Advant® OCS
with Master software

AMPL Application Building

Reference Manual

Use of **DANGER**, **WARNING**, **CAUTION**, and **NOTE**

This publication includes, **DANGER**, **WARNING**, **CAUTION**, and **NOTE** information where appropriate to point out safety related or other important information.

DANGER	Hazards which could result in severe personal injury or death
WARNING	Hazards which could result in personal injury
CAUTION	Hazards which could result in equipment or property damage
NOTE	Alerts user to pertinent facts and conditions

Although **DANGER** and **WARNING** hazards are related to personal injury, and **CAUTION** hazards are associated with equipment or property damage, it should be understood that operation of damaged equipment could, under certain operational conditions, result in degraded process performance leading to personal injury or death. Therefore, comply fully with all **DANGER**, **WARNING**, and **CAUTION** notices.

TRADEMARKS

ABB MasterAid is registered trademark of ABB Asea Brown Boveri Ltd. Switzerland.

Advant is a registered trademark of ABB Process Automation Corp., USA.

Advant Controller is a trademark of ABB Process Automation Corp., USA.

NOTICE

The information in this document is subject to change without notice and should not be construed as a commitment by ABB Industrial Systems AB. ABB Industrial Systems AB assumes no responsibility for any errors that may appear in this document.

In no event shall ABB Industrial Systems AB be liable for direct, indirect, special, incidental or consequential damages of any nature or kind arising from the use of this document, nor shall ABB Industrial Systems AB be liable for incidental or consequential damages arising from use of any software or hardware described in this document.

This document and parts thereof must not be reproduced or copied without ABB Industrial Systems AB's written permission, and the contents thereof must not be imparted to a third party nor be used for any unauthorized purpose.

The software described in this document is furnished under a license and may be used, copied, or disclosed only in accordance with the terms of such license.

Copyright © ABB Industrial Systems AB 1996

3BSE001264/C

TABLE OF CONTENTS

Chapter 1 - Introduction

1.1 General 1-1
1.2 Designing with AMPL 1-3
1.3 Related Documents 1-5

Chapter 2 - General

2.1 General 2-1
2.2 PC Elements 2-3
2.3 PC Programming 2-4
2.4 Program Modules 2-5
2.5 Identification 2-6
2.6 Execution 2-6
2.7 Final Draft 2-7
2.8 Program Entry and Documentation 2-9

Chapter 3 - Data Base Elements

3.1 General 3-1
3.2 Graphic Symbol of the Data Base Element 3-2
3.3 Data Base Element Header 3-3
3.4 NAME terminal 3-4
3.5 Dimensioning 3-5
3.6 Entering Data into the Data Base Element 3-5
3.7 Connections 3-5

Chapter 4 - ABB Master Programming Language - AMPL

4.1 Data Types 4-1
    4.1.1 Boolean 4-2
    4.1.2 Integer 4-3
    4.1.3 Real Numbers 4-5
    4.1.4 Time 4-7
    4.1.5 Array 4-9
    4.1.6 Group data 4-10
4.2 PC Elements 4-12
4.3 Call Names for PC Elements 4-13
4.4 Call Parameters for PC Elements 4-14
    4.4.1 Size Parameters 4-14
    4.4.2 Data Type Parameters 4-18
    4.4.3 Variant-determining Call Parameters 4-19
    4.4.4 Call parameters for structure elements 4-19

CONTENTS (continued)

4.5 PC Element Terminals 4-19
  4.5.1 Terminal Numbers 4-20
  4.5.2 Terminal Names 4-21
  4.5.3 Terminal Type 4-22
  4.5.4 Dynamic Inputs and Inverted Signals 4-23
4.6 Symbol Types 4-23
4.7 Variables and Constants 4-26
  4.7.1 Output Variables 4-26
  4.7.2 Connecting to Data Base Elements 4-27
  4.7.3 Operational Parameters 4-27
  4.7.4 Constants 4-28
  4.7.5 Multiple Use of Operational Parameters and Constants 4-28
  4.7.6 Function Parameters 4-29
4.8 Symbolic Naming of Variables and Constants 4-30
  4.8.1 Output Variables 4-30
  4.8.2 Operational Parameters and Constants 4-30

Chapter 5 - Structure Elements of the PC Language

5.1 Dividing the PC Program into Functional Sections 5-1
5.2 Execution Units 5-5
5.3 Hierarchy of the Module Headers 5-7

Chapter 6 - Function Modules- FUNCM

6.1 General 6-1

Chapter 7 - Control Modules - CONTRM

7.1 Call 7-1
7.2 Control 7-2
7.3 RUN-indication 7-3
7.4 Blocking from the Engineering Station 7-4
7.5 Modify Permission 7-4
7.6 Execution of a Control Module 7-5
7.7 Control Module in the Hierarchy 7-5

Chapter 8 - PC Programs in Advant Controller 400 Series

8.1 General 8-1
8.2 Call 8-2
8.3 Control 8-2
8.4 Blocking from the Engineering Station 8-3

CONTENTS (continued)

Chapter 9 - Master-Slave Combination

9.1 General 9-1
9.2 Master Header - MASTER 9-2
    9.2.1 Call 9-2
    9.2.2 Control 9-2
9.3 Slave Module - SLAVEM 9-3
    9.3.1 Call 9-3
9.4 Blocking from the Engineering Station 9-3
9.5 Modify Permission 9-4
9.6 Master-slave in the Hierarchy 9-4

Chapter 10 - Sequence and Step Modules

10.1 General 10-1
10.2 Dividing of a Sequence into Steps 10-4
10.3 Sequence in the Hierarchy 10-4
10.4 Function 10-5
    10.4.1 RUN Indications 10-8
    10.4.2 Step Module Input and Output 10-8
10.5 Sequence Header- SEQ 10-10
    10.5.1 Call 10-10
    10.5.2 Control 10-11
    10.5.3 Blocking from the Engineering Station 10-12
    10.5.4 Indications 10-12
    10.5.5 Resetting a Sequence 10-13
    10.5.6 Manual Control 10-14
    10.5.7 HOLD 10-15
    10.5.8 Time Monitoring 10-16
    10.5.9 Signals for Operator Station Functions 10-17
10.6 Step Module - STEP 10-18
    10.6.1 Call 10-18
    10.6.2 Condition Input (COND) 10-19
    10.6.3 RUN Indication 10-19
    10.6.4 Time Function 10-20
    10.6.5 Conditional Jump 10-21
    10.6.6 Selective Reset 10-22

Chapter 11 - PC Block - BLOCK

11.1 General 11-1

CONTENTS (continued)

Chapter 12 - The PC Element Types

12.1 General 12-1

Chapter 13 - PC Programming

13.1 Identification (Item Designation) of PC Elements 13-1
13.2 Identification of PC Element Terminals 13-4
13.3 Connection of PC Elements 13-4
  13.3.1 Connection to Data Base Element 13-11
  13.3.2 Connection to Operational Parameters 13-12
  13.3.3 Connection to Constants 13-13
  13.3.4 Interconnection of PC Programs 13-14
13.4 Preparing PC Diagrams 13-16
  13.4.1 Page Numbering 13-18
  13.4.2 Printout of PC Diagram 13-18

Chapter 14 - Execution

14.1 PC Programs 14-1
14.2 Interpreter 14-2
14.3 Kernel 14-2
14.4 Process l/O Handling 14-3
14.5 Cycle Time Table 14-4
14.6 Several Interpreters 14-5
14.7 Input and Output Phases 14-5
  14.7.1 Input 14-6
  14.7.2 Output 14-6
14.8 Execution Order 14-8
14.9 Execution of PC Elements 14-9
14.10 Reset Execution 14-9
14.11 Initialization 14-10

Chapter 15 - Applications with Sequences

15.1 Function Diagram 15-1
15.2 Activities 15-3
  15.2.1 Continuous Order (C) 15-4
  15.2.2 Time Controlled Order (T) 15-5
  15.2.3 Activities with Memory Function (S) 15-7
15.3 Single Sequences 15-9
15.4 Beginning of Sequence Selection (Divergence) 15-12

CONTENTS (continued)

15.5 End of Sequence Selection (Convergence) 15-16
15.6 Complete Sequence Selections 15-17
15.7 Sequence Selection with Several Terminating Branches 15-20
15.8 Simultaneous Sequences 15-21
15.9 Sequence with Loop 15-23
15.10 Use of HOLD 15-25

Chapter 16 - Application Example

16.1 Control of Lift 16-1

Chapter 1 Introduction

1.1 General

Advant Controller 400 Series is a system for digital and feedback control of processes of a wide range of size and complexity. For easy handling the following programs, methods and tools are available:

- A programming language, ABB Master Programming Language (AMPL), which is a process-oriented high level language that uses graphic symbols.
- Methods for control system development from specification to complete system and maintenance.
- Engineering tools for use at different levels in the procedures included in the programming, testing, documentation maintenance and debugging.

This manual describes the AMPL and its application in a general manner. Limitations in the use of the language in any particular system are specified in the appropriate chapter.

The manual has been divided into one general introduction, comprising Chapter 1 and Chapter 2, and two reference sections, the first of which describes the building blocks of the language and the second describe the assembly and application of these building blocks.

In part 1 the Data Base Elements (Chapter 3), the AMPL (Chapter 4), the structure PC elements (Chapter 5 - Chapter 11) and the function PC element (Chapter 12) are described.

Part 2 describes the practical application of the language. It describes how a program is prepared and how the execution can be managed. Finally instructions for the assembly of sequences and some application examples are given.

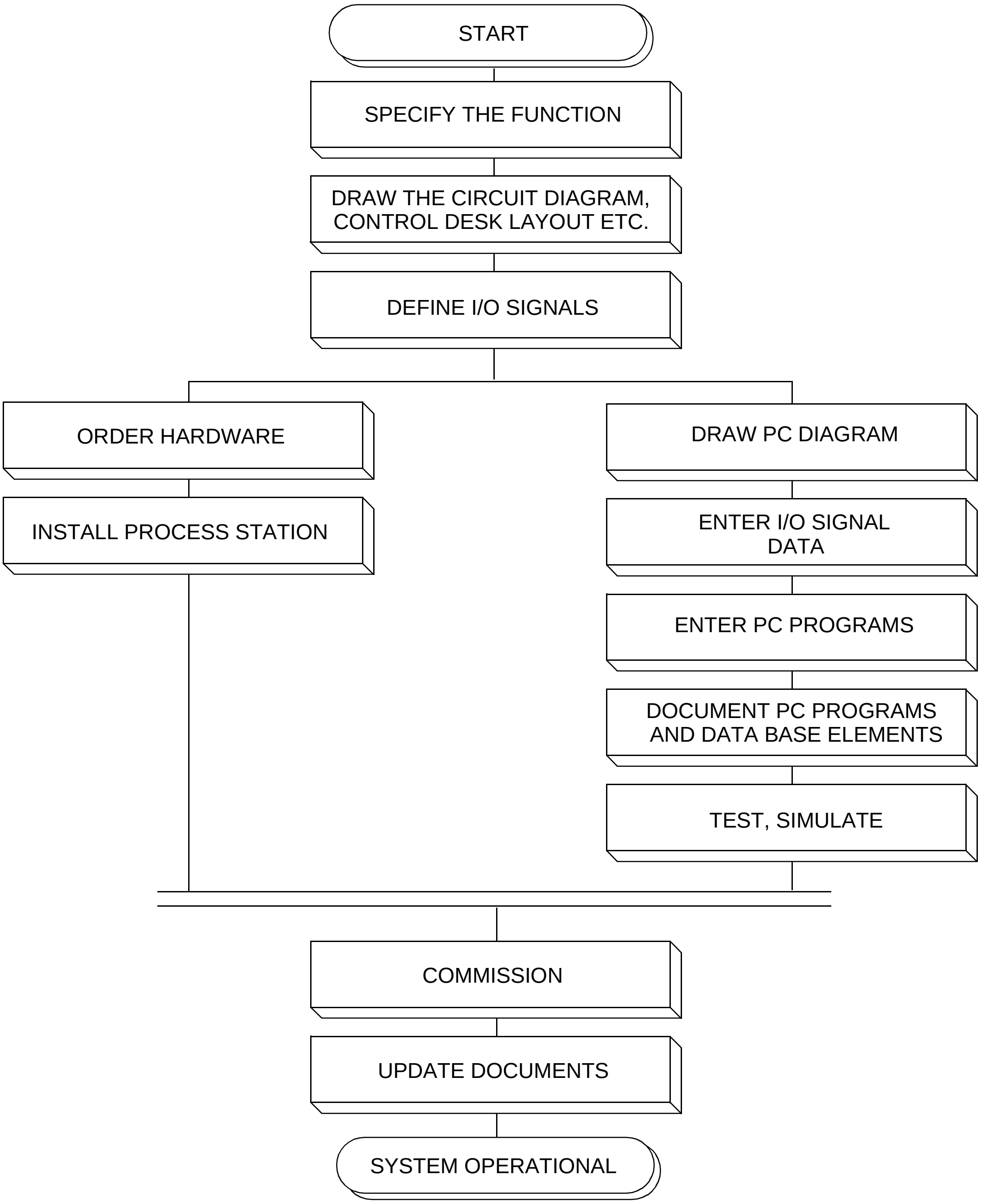


Figure 1-1. Flow Diagram for Designing with AMPL

1.2 Designing with AMPL

The flow diagram in Figure 1-1 shows the procedure to be followed in designing an installation. The scope of the control equipment is specified first. The arrangement of the I/O channels is then determined and the plant circuit diagram can be drawn. The work can then be divided in a hardware part, the ordering of the necessary equipment and so forth, and a software part, the development of the programs.

- The definition of the input and output signals will result in parameters for the Data Base elements and is described in the design manual for the product used. In Chapter 3 of this manual the structure and use of the Data Base elements is described.
- The software part begins with the preparation of a draft PC diagram. This is used as a basis for the input data to the Engineering Station. This manual describes the appearance of the draft and the rules which apply in its preparation.
- The entry of the information from the two preceding steps in the procedure and the processing of this information to generate a finished program for the control equipment is described in the manual for the Engineering Station that is used.
- A printer can be connected to the Engineering Station to document the program in the form of a PC diagram and documents for the Data Base elements.

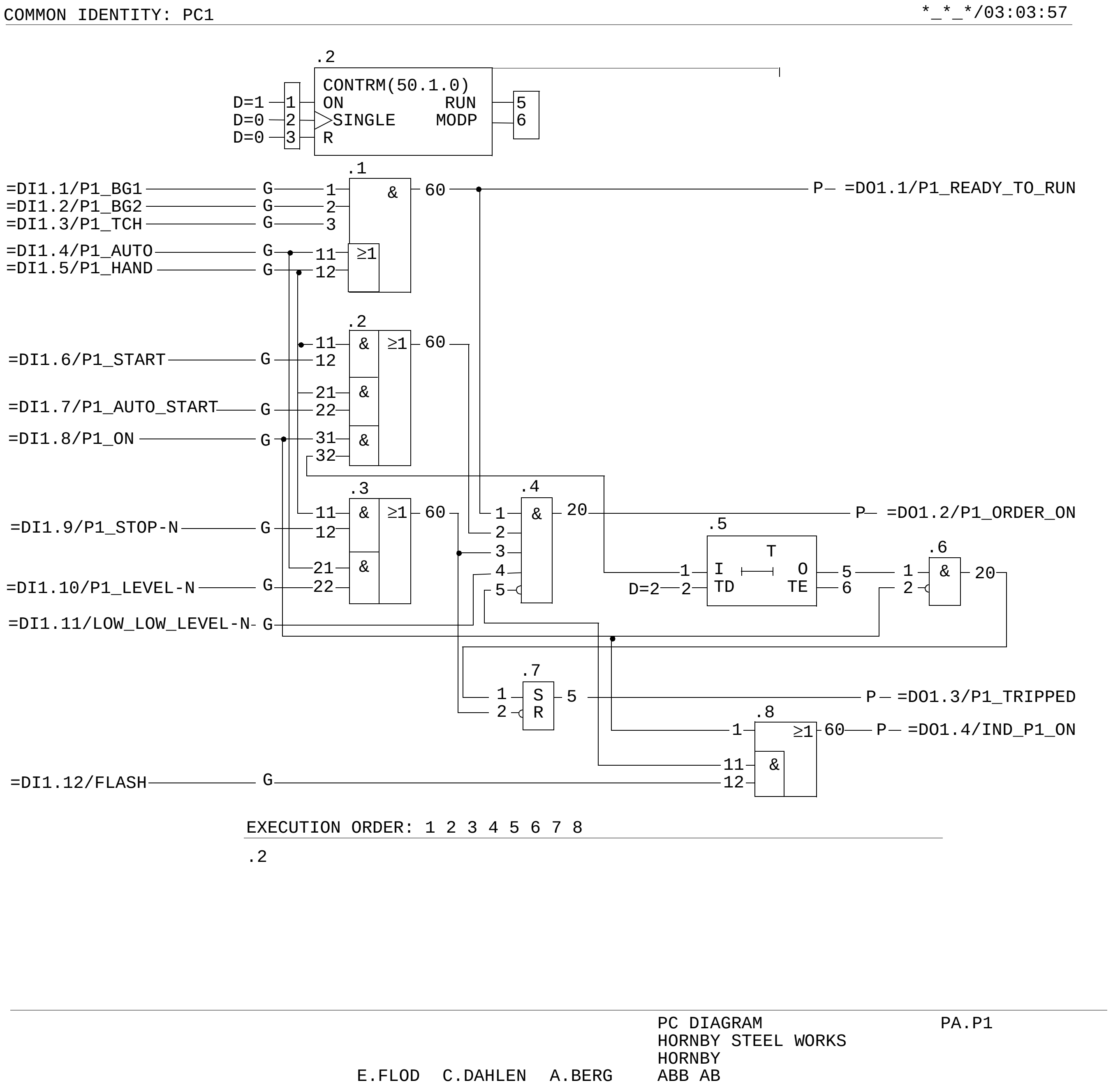


Figure 1-2. Example of PC Diagram

1.3 Related Documents

The table below lists documents which may be of use to the AMPL programmer.

Title

- *Advant Controller 410 Series User’s Guide*
- *Advant Controller 450 Series User’s Guide*
- *PC Elements Advant Controller 400 Series Reference Manual*
- *Data Base Elements Advant Controller 400 Series Reference Manual*
- *AMPL Configuration Advant Controller 400 Series Reference Manual*
- *MasterAid 220 User´s Manual*

Chapter 2 General

2.1 General

The introduction above outlined the working procedure in designing with AMPL.

The following is a simple example to illustrate the development of a PC program.

Figure 2-1 is a circuit diagram for a pump unit with motor, start switch, ON-OFF buttons and operation indicator lamp. It is assumed that the control function is performed with a programmable controller system symbolized in the circuit diagram by a block (+ D2). The block gives a reference to a corresponding section in a PC program.

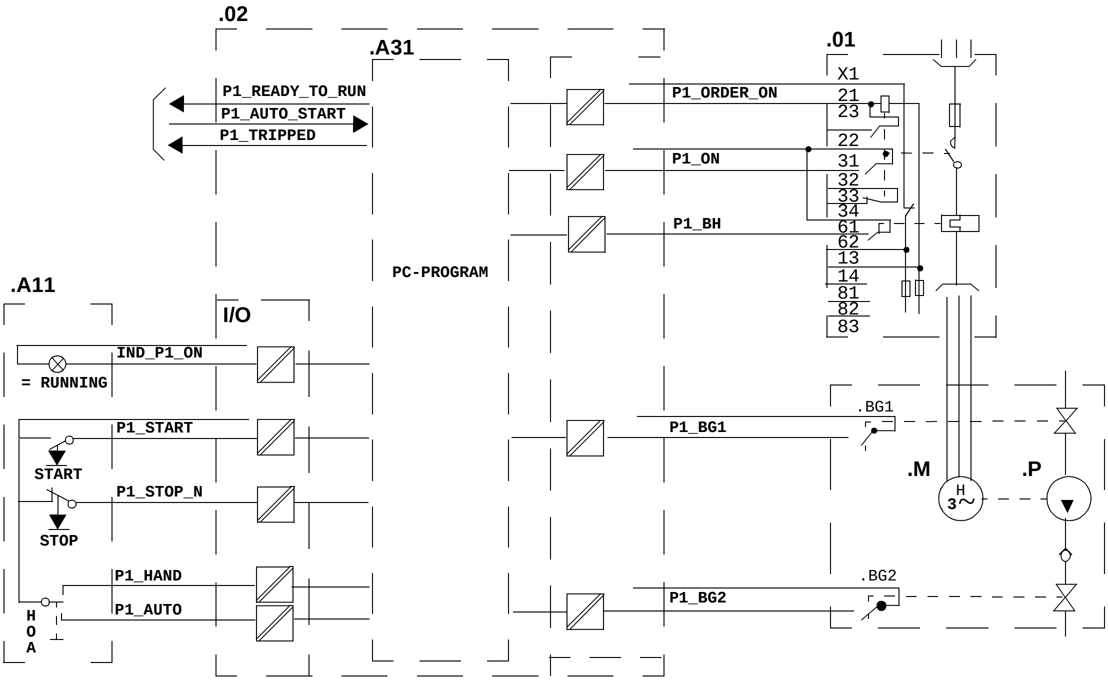


Figure 2-1. Example - Control of Pump Unit Motor

The control function is then drawn using logic symbols as shown in Figure 2-2. Each such symbol (AND-gate, timer, flip-flop, and so forth) is, in the programmable controller system, called a PC element.

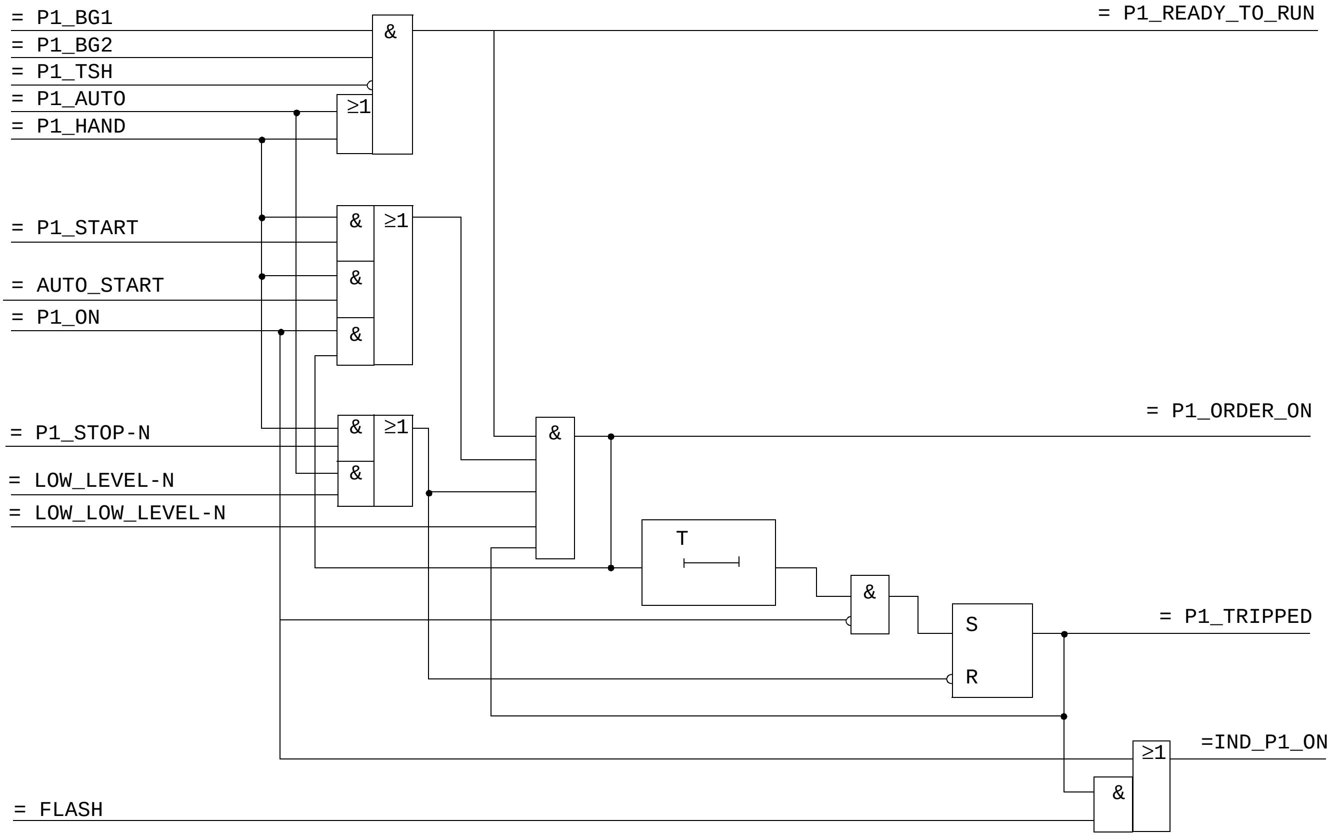


Figure 2-2. Control Function in PC Diagram Form for the Pump Motor in Figure 2-1

2.2 PC Elements

The basic unit of AMPL is the PC element. Each PC element implements a complete function such as a counter, a delay PC element or an analog controller. Each PC element is represented in a PC diagram by a graphic symbol designed to describe the function of the PC element.

Figure 2-3 provides a comparison between the symbols for some common PC elements and the corresponding relay symbols.

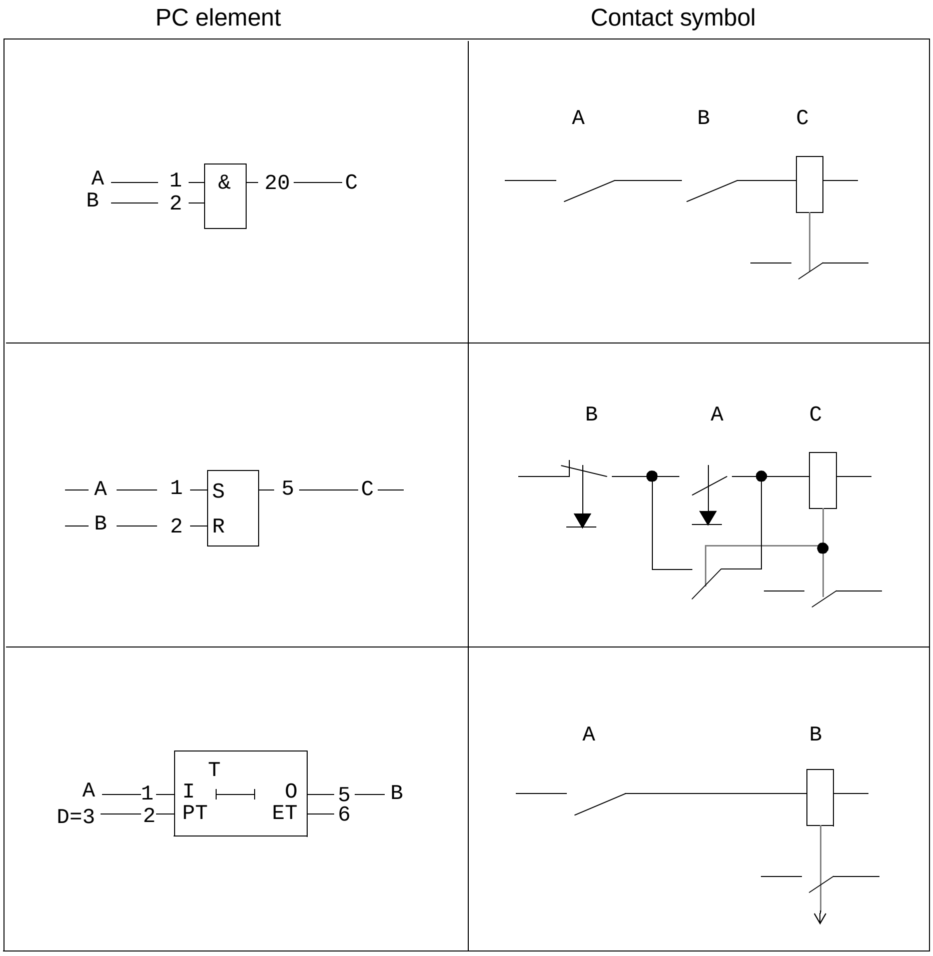


Figure 2-3. Comparison between PC Elements and their Corresponding Relay Symbols

2.3 PC Programming

PC programming is performed as follows:

- Suitable PC elements are first selected from a specified range and the corresponding symbols are entered in a draft diagram. The range of PC elements, that is the PC element library, is selected to suit the process function.
- The PC elements are then assembled to provide the required function.

The completed diagram is designated the PC diagram and resembles, in principle, a traditional logic and control diagram (Figure 2-4).

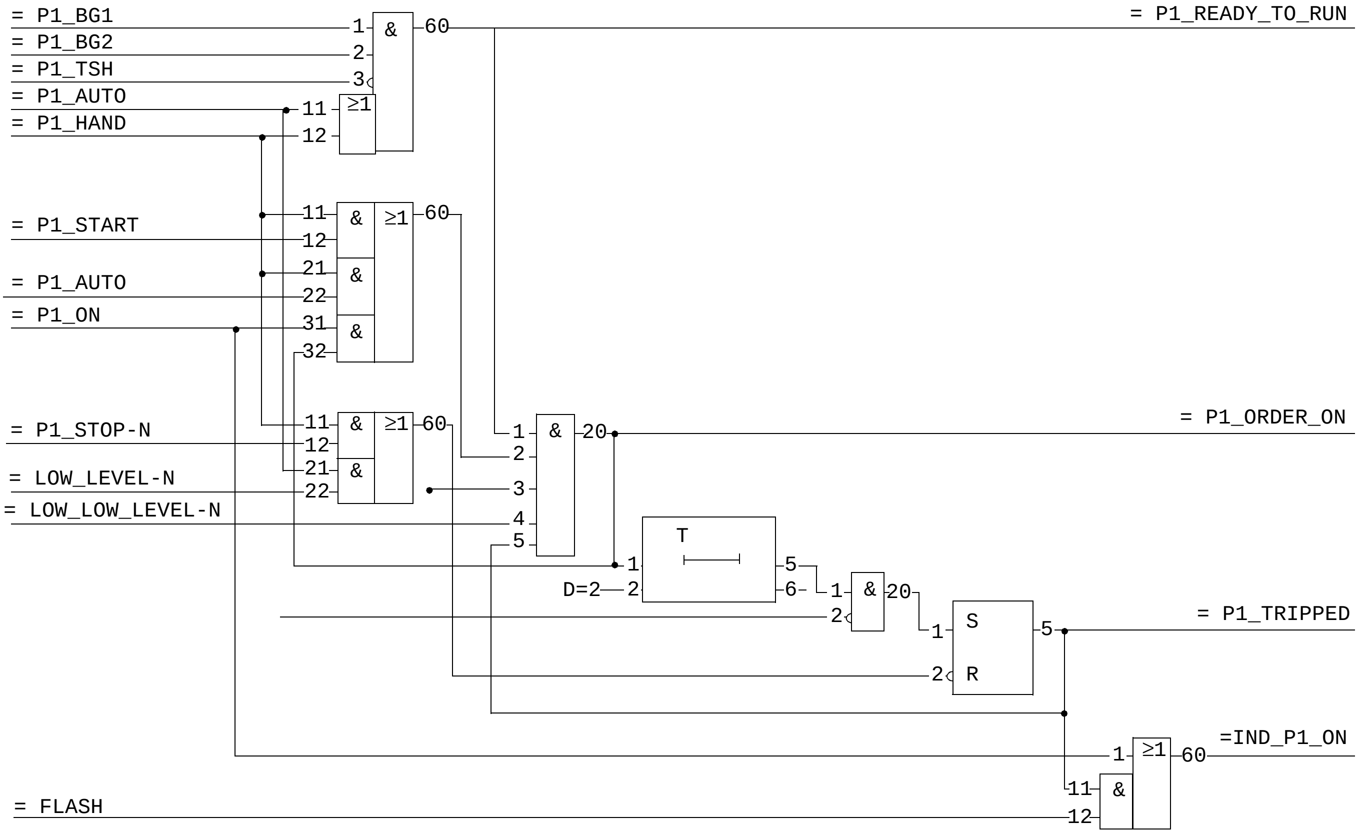


Figure 2-4. Draft PC Diagram

2.4 Program Modules

A PC program can be divided into several program modules. This separation or structuring should be done in such a way that it reflects the structure of the process or plant.

The structure may consist of several levels in an hierarchic system. A small system for control of a single machine may have only one level, whereas a large system for control of a power station or a rolling mill may have several levels.

The example of pump motor control may be expanded by the addition of a second pump unit and a co-ordinated control of the two units, as shown in Figure 2-5.

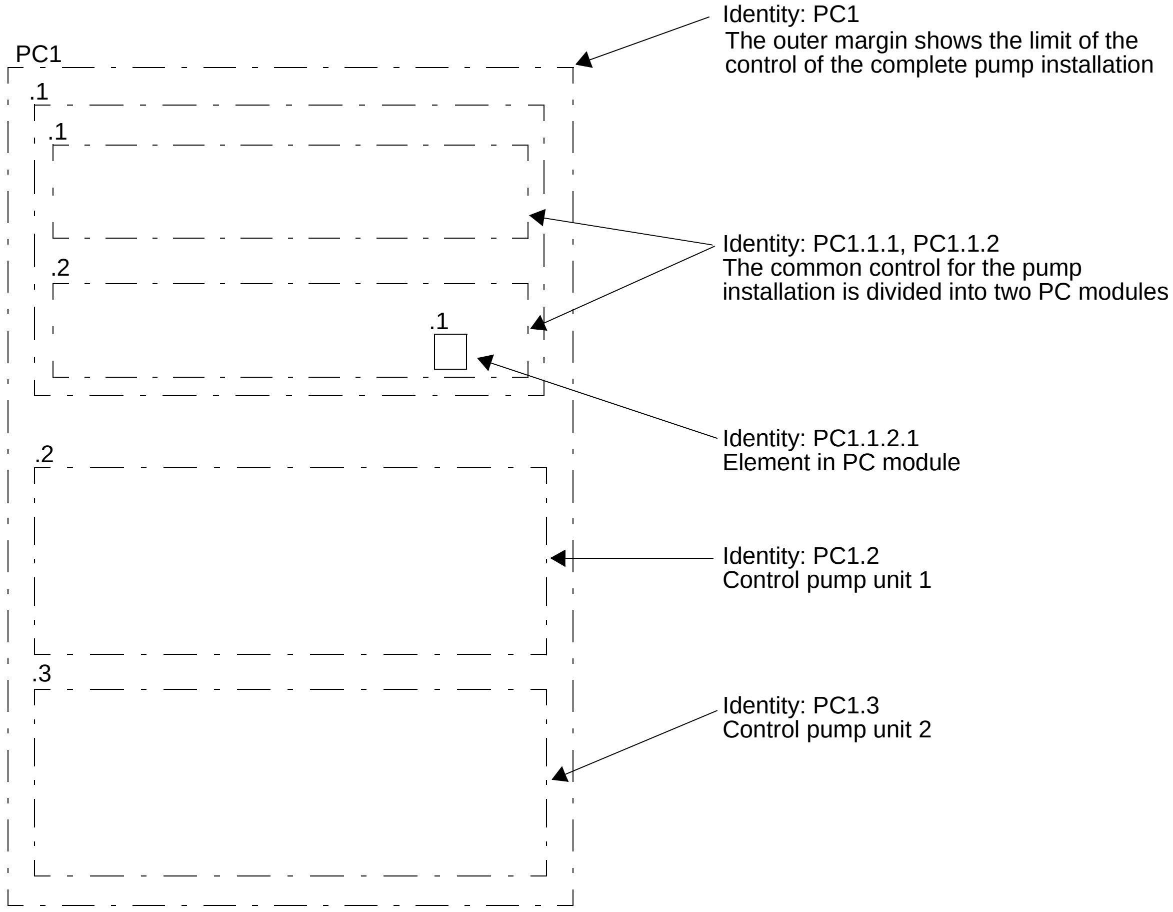

PC1
.1
.1
.2
.1
.2
.3
Identity: PC1
The outer margin shows the limit of the control of the complete pump installation
Identity: PC1.1.1, PC1.1.2
The common control for the pump installation is divided into two PC modules
Identity: PC1.1.2.1
Element in PC module
Identity: PC1.2
Control pump unit 1
Identity: PC1.3
Control pump unit 2

Figure 2-5. Structuring of a PC Program

2.5 Identification

Figure 2-5 suggests an item designation system for identification of PC modules and PC elements. The system has an hierarchic structure similar to the division of a document into chapters and sections. In the example the PC element in the subsection for pump control has the designation P C 1.1.2.1. The item designations are used in program entry and in the final program documentation and are also utilized as search references when testing and tracing faults in the program.

2.6 Execution

A PC program is always executed cyclically, PC element by PC element, within each PC module. The time between the starts of two consecutive cycles is called cyclicity.

The cyclicity can be set individually for each PC module. For this purpose each module is provided with a module header, which is located in the upper left corner in the PC diagram.

The module headers have, amongst others, the inputs ON and Reset to switch the execution of the module on and off.

The module headers are described in more detail in Chapter 5 - Chapter 11 of this manual. Figure 2-6 and Figure 2-7 show two examples of module headers, one (PCPGM) is intended for the highest PC program level, the other (CONTRM) for ordinary PC modules.

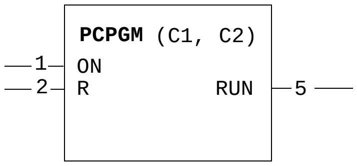


Figure 2-6. PCPGM

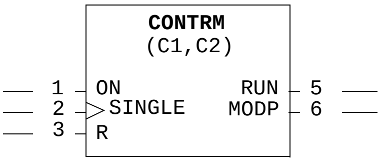


Figure 2-7. CONTRM

The two main purposes of dividing the program into modules are:

- To obtain a more easily surveyed program documentation which in its arrangement agrees with the other plant documentation.
- To allow the execution of each individual program module to be controlled according to its own requirements. The execution of each module may also be inhibited, for example to select manual or automatic operation.

The execution may also be switched on and off by means of a service tool connected to the equipment.

2.7 Final Draft

Figure 2-8 shows the example from Figure 2-5 with the PC modules supplemented with module headers.

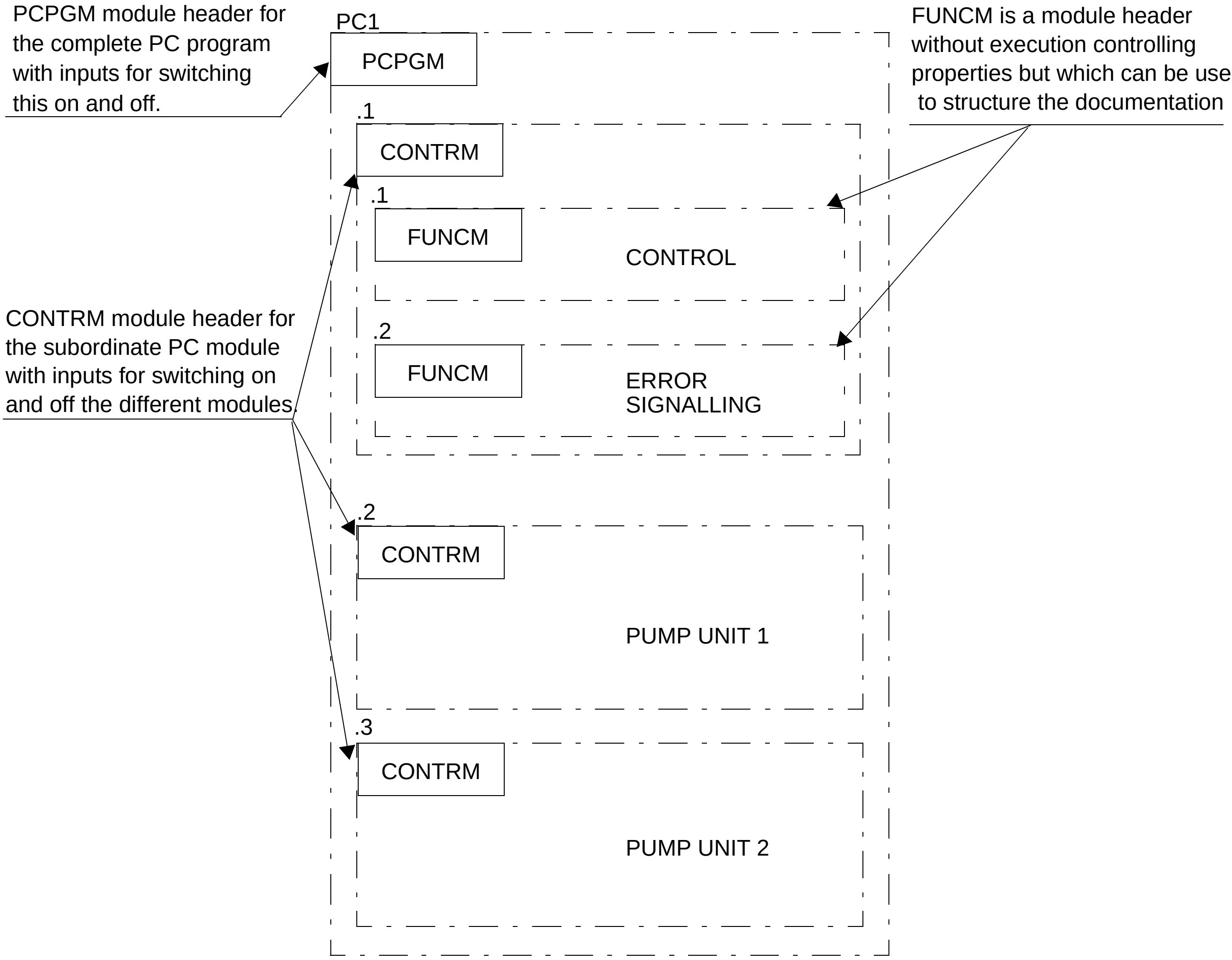

PCPGM module header for the complete PC program with inputs for switching this on and off.
PC1
PCPGM
FUNCM is a module header without execution controlling properties but which can be used to structure the documentation
.1
CONTRM
.1
FUNCM
CONTROL
CONTRM module header for the subordinate PC module with inputs for switching on and off the different modules
.2
FUNCM
ERROR SIGNALLING
.2
CONTRM
PUMP UNIT 1
.3
CONTRM
PUMP UNIT 2

Figure 2-8. Structuring of a PC Program with Module Headers

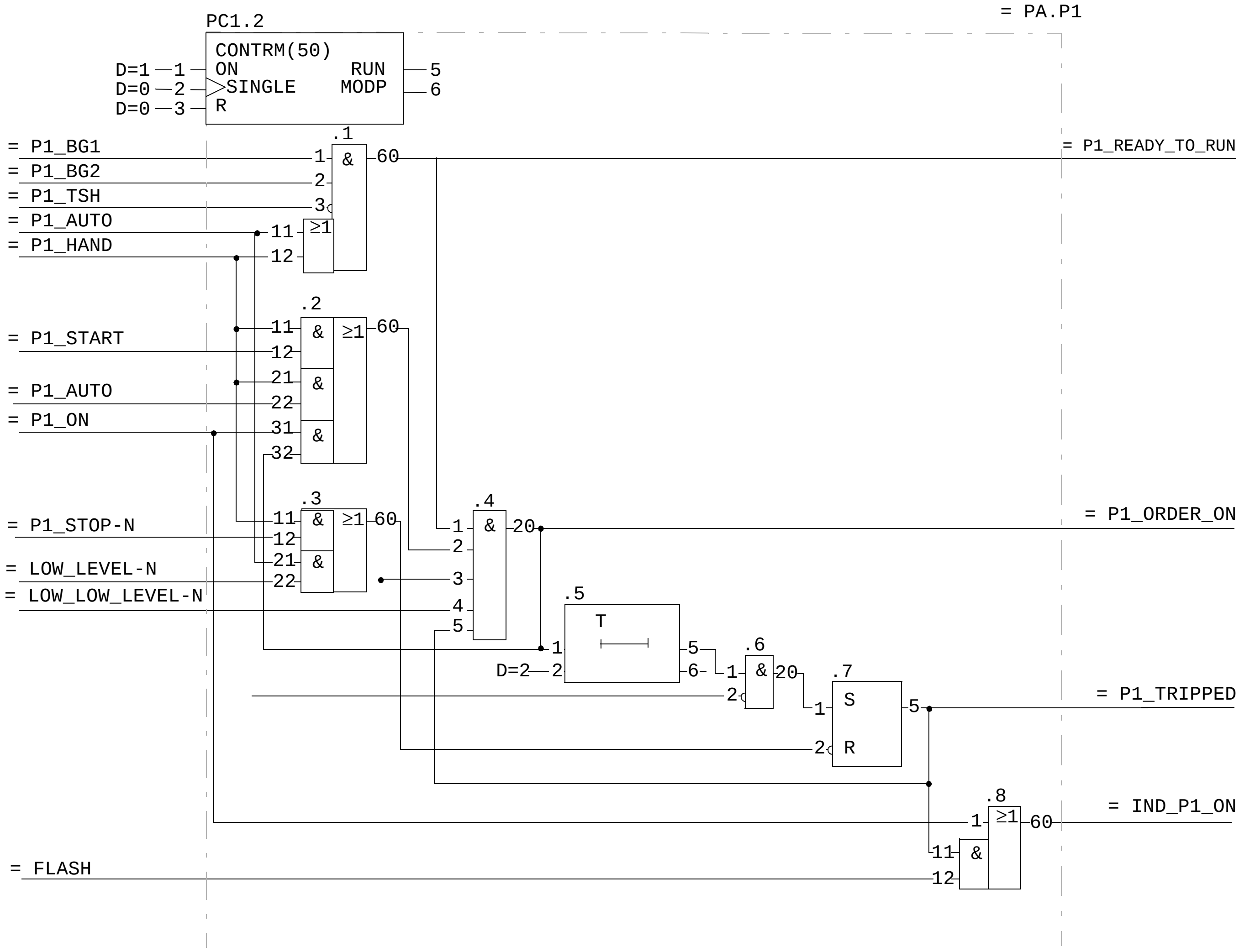


Figure 2-9. Final Draft of PC Diagram for Module PC1.2

Figure 2-9 shows the final draft PC diagram for the motor control operation, module PC1.2.

This draft PC diagram constitutes a basis for program entry.

The program entry is performed with a programming tool which translates the information entered to internal object code for the Advant Controller 400 Series system.

2.8 Program Entry and Documentation

Entering the information from the draft is normally done by the programmer conducting a dialogue with a programming tool using its keyboard and screen. This dialogue is described in detail in the manual for the programming tool.

The program entry is done in two steps:

1. Name and parameters for the PC elements to be used are entered.
2. PC elements are connected to each other and to the Data Base elements.

Documentation can be obtained from the Engineering Station when these two steps are finished.

Using a graphic printer a PC diagram in various formats with drawing header and text can be obtained.

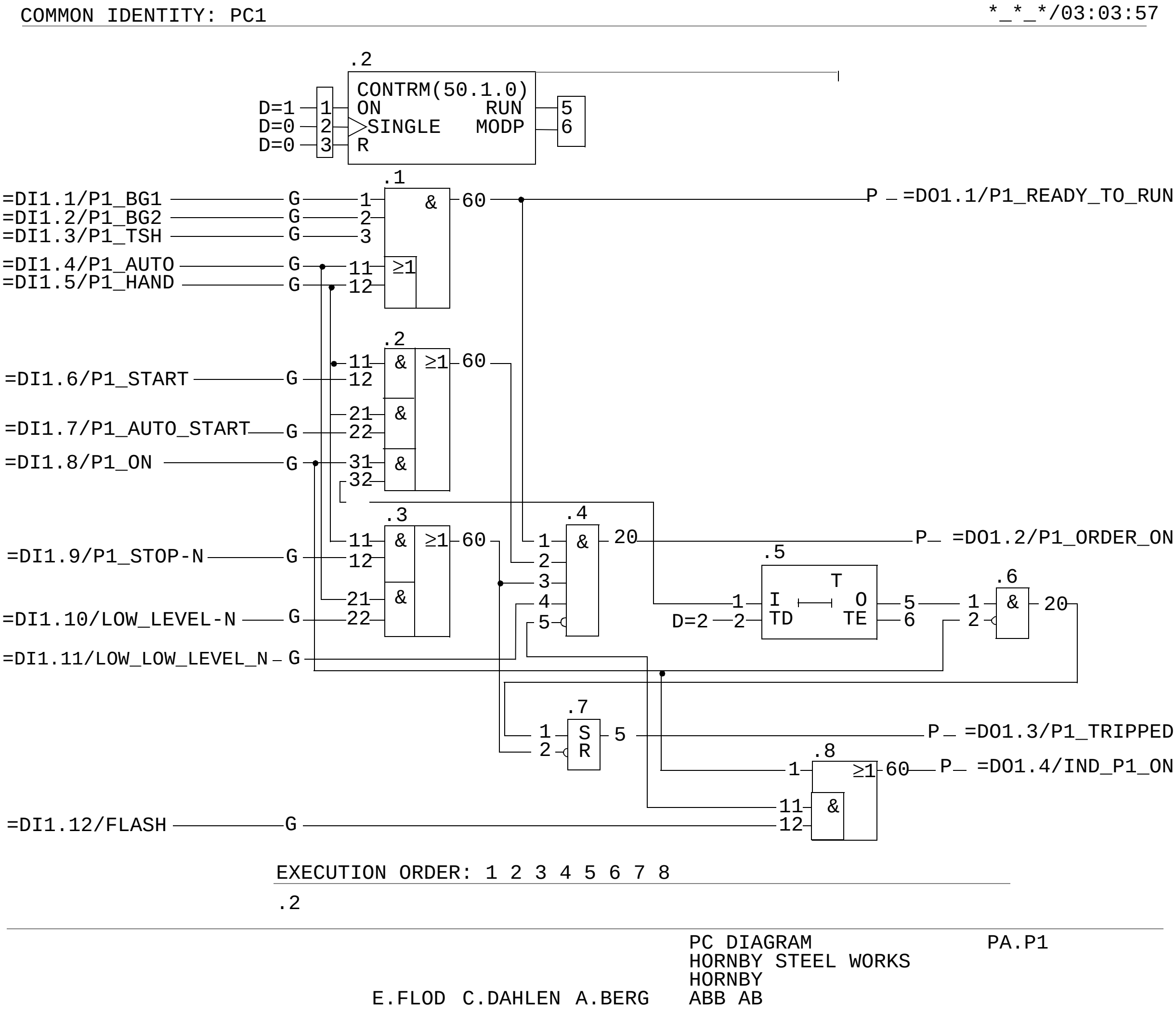


Figure 2-10. PC Diagram Printout

Chapter 3 Data Base Elements

3.1 General

A Data Base is used by the PC programs to exchange data with their environment. This Data Base is accessible to different types of programs such as Operator Station programs, I/O communication and computer communications programs. The Data Base uses a standardized storage method to be easily accessible to different programs.

The Data Base is also suitable for storage of hardware dependent information like board addresses, communication networks and so on.

In addition, the Data Base is used for data exchange between PC programs.

The layout and function of the Data Base for the Advant Controller 400 Series systems are described in detail in the *Data Base Elements Advant Controller 400 Series Reference Manual.*

The size of the different constituent parts of the Data Base is defined using the dimensioning command of the Engineering Station. When this has been done, the user may easily enter the values required to give the Data Base its proper contents.

Data Base Elements for Advant Controller 400 Series

Data Base elements are used to enter, modify and view the Data Base contents.

Each one of these elements contains the information necessary for a complete description of, for example, an analog input or set-up data for a printer.

For the I/O system, each board is represented by a Data Base element. Each channel of the board is in turn represented by its own Data Base element. Examples of board and channel elements are shown in Figure 3-1 and Figure 3-2 respectively.

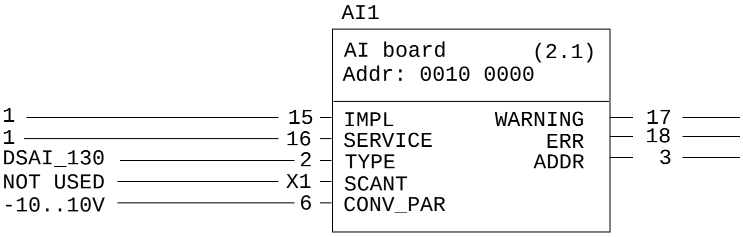


Figure 3-1. Data Base Element for an AI Board

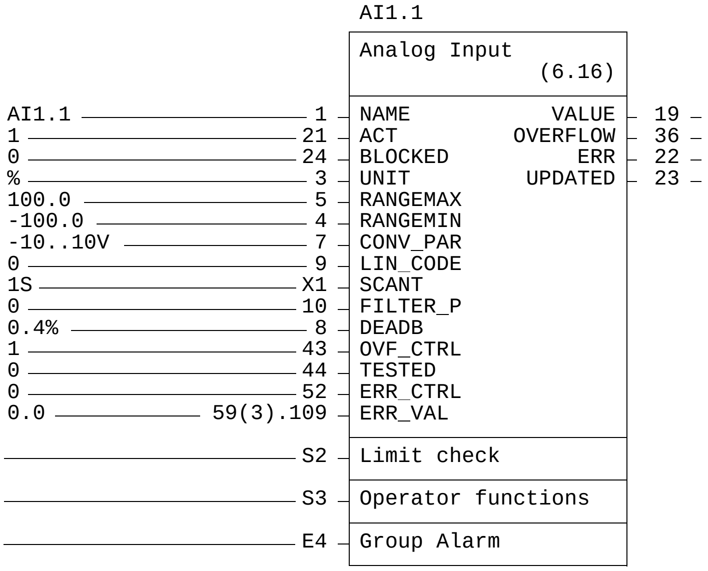


Figure 3-2. Data Base Element for an AI Channel

3.2 Graphic Symbol of the Data Base Element

Each Data Base element has a graphic symbol. For large elements the symbol is subdivided into a number of segments, S1, S2, ..., Sn.

When an element is displayed on the screen, the header, one expanded segment and the other segments in compressed form are shown.

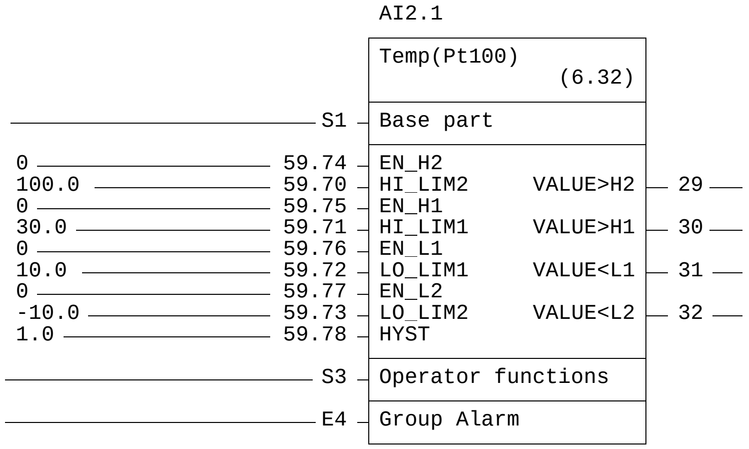


Figure 3-3. Graphic Symbol for a Data Base Element

The segments of the symbol have one terminal for each data item. Input data are shown on the left and output data on the right hand side of the element. The names inside the symbol are terminal names that identifies the data items. These names are, for example used when connecting PC elements to Data Base elements. Immediately outside the symbol the internal number (so called property number) of the data item is given.

3.3 Data Base Element Header

The call name is a mnemonic name for the element type and is usually the first part of the item designation (See below). For a small number of elements the call name is followed by a call parameter for determination of the data type.

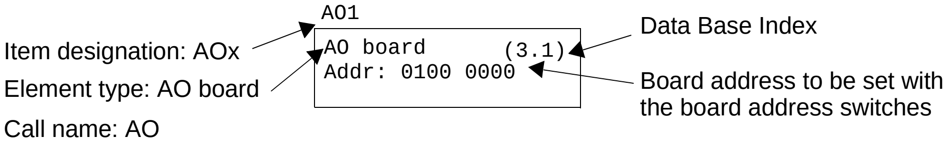


Figure 3-4. Data Base Element Header

The PC element type is described on the first line in the PC element header.

The Data Base elements are identified hierarchically with an item designation based on, for example, board number and channel number.

Example:

DI3	denotes the 3rd DI board
AO8	denotes the 8th AO board
DI3.2	denotes the 2nd channel of board DI3

Most elements may, as an alternative identification, be given a unique user defined name (terminal 1, NAME). If no such name is given the item designation is used.

The Data Base element index (internal numbers for data access) may also be used to identify the element.

For board elements the element header shows the required setting of the address straps.

For communications a corresponding subdivision called data set is used. A Data Base element is defined for each data set and a number of Data Base elements, one for each data, are defined for the data in the data set.

Data Base elements are further used to define parameters for communication, printers, weighing systems, and so forth.

Table 3-1. Examples of Data Base Elements

Item Designation	Call Name	Element Type
AIx.y	AI	Analog input
AIx.y	AIPT	Temp (Pt100)
AIX.y	AITC	Thermocouple
DIX.y	DI	Digital input
DIx.y	PULSE	Pulse board
CAPx.y	EXCOM	External communication
CAPx.y	PRINT	Printer set-up
DATx	DAT	Boolean data
DATx	DAT	Integer data
DATx	DAT	Integer long data
DATx	DAT	Real data
GENx	CON	PI controller
GENx	BIN	Motor/Value ctrl
GENx	USD	User defined ctrl

3.4 NAME terminal

For those Data Base elements that have a terminal for NAME a user defined name may be given. This name may then be used as a reference in PC programming, when changing Data Base elements and so forth.

In the picture of a Data Base element, the name is written after the item designation. The name and the item designation are separated by a slash (“/”).

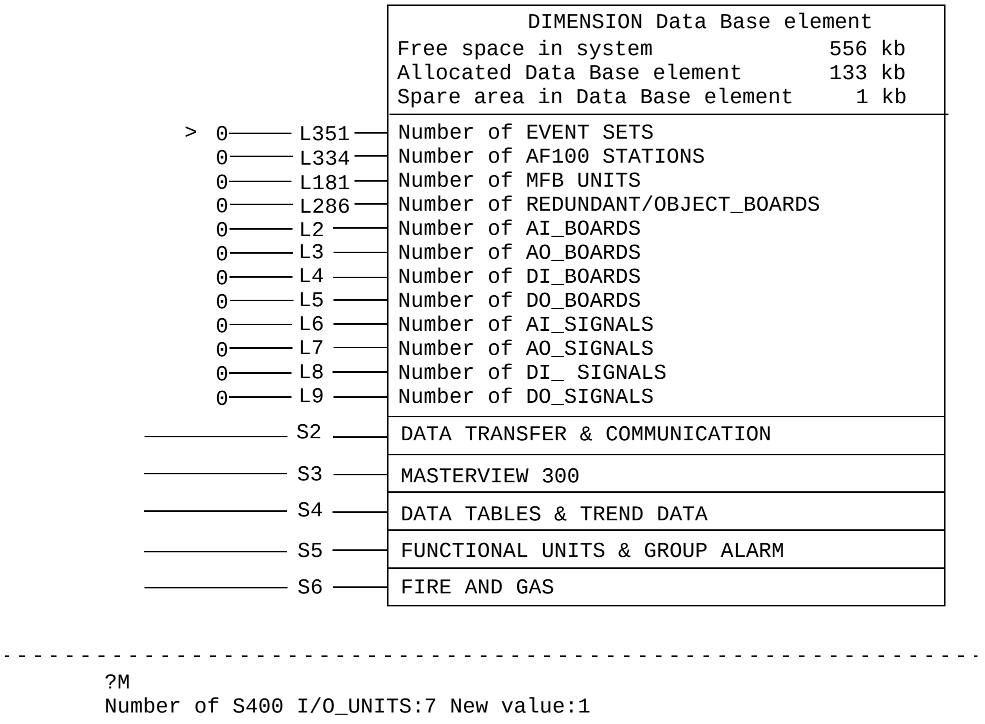
The name may be up to 20 characters in length and may contain most characters with the exception of space. In the Operator Station the name is truncated to 12 characters. Names that may be interpreted as an item designation are not allowed if it is not the exact item designation of the element.

In the manual for *Data Base Elements Advant Controller 400 Series* and *PC Elements Advant Controller 400 Series* a description in detail is given for each Data Base element and PC elements, respectively.

3.5 Dimensioning

The handling of Data Base elements is supported by the Engineering Station with powerful functions for definition of the Data Base element contents.

A dimensioning function is used to achieve an optimal organization of the memory for PC and Data Base element. Using this function the number of boards and channels for each main type (AI, DI, and so on), the number of data sets and so forth can be selected. Figure 3-5 shows the form used in dimensioning.



```
                        DIMENSION Data Base element
                   Free space in system              556 kb
                   Allocated Data Base element       133 kb
                   Spare area in Data Base element     1 kb
 > 0   L351        Number of EVENT SETS
   0   L334        Number of AF100 STATIONS
   0   L181        Number of MFB UNITS
   0   L286        Number of REDUNDANT/OBJECT_BOARDS
   0   L2          Number of AI_BOARDS
   0   L3          Number of AO_BOARDS
   0   L4          Number of DI_BOARDS
   0   L5          Number of DO_BOARDS
   0   L6          Number of AI_SIGNALS
   0   L7          Number of AO_SIGNALS
   0   L8          Number of DI_ SIGNALS
   0   L9          Number of DO_SIGNALS
       S2          DATA TRANSFER & COMMUNICATION
       S3          MASTERVIEW 300
       S4          DATA TABLES & TREND DATA
       S5          FUNCTIONAL UNITS & GROUP ALARM
       S6          FIRE AND GAS
-------------------------------------------------------------
        ?M
        Number of S400 I/O_UNITS:7 New value:1
```

Figure 3-5. DIMDB, Modify Mode

3.6 Entering Data into the Data Base Element

Following dimensioning, the Data Base element contents must be entered by the user. During entering the Data Base elements are shown on the screen of the Engineering Station with default values. These defaults are chosen so that they seldom has to be changed.

For an I/O board all data are automatically entered for the board element as well as for all the subordinate channel elements when the board type has been entered.

3.7 Connections

See Section 4.7.2 Connecting to Data Base Elements for information on connection between PC and Data Base element.

Chapter 4 ABB Master Programming Language - AMPL

The PC element is the basic unit by which the user composes his program. A PC element is either a function element representing, for example, a logical or an arithmetic function or a structure element in the form of a module header.

A PC element is represented in the PC-diagram by a graphic symbol with terminals for input and output variables.

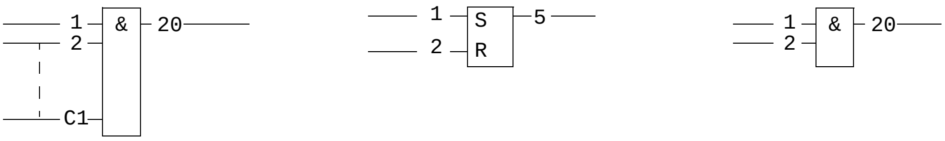


Figure 4-1. Examples of Graphic Symbols for PC Diagrams

4.1 Data Types

PC elements are capable of handling data of different types such as logical (Boolean) variables, integers, real numbers and so forth.

A number of data types with specific properties and range of values have been defined. The data exchange between the PC elements and with the Data Base element has thus been standardized. A further result is that this permits an automatic validity check of the format and values of data during program entry.

The following data types are used in Advant Controller 400 Series systems:

B	Boolean (logical 1 or 0)
I	Integer, 16 bit
IL	Integer, 32 bit
R	Real numbers
T	Time
TR	Time expressed as a real number
A	Array, for example a character string
G	Group data (several data of the same data type)

If the result of a calculation within a PC element exceeds the legal range for the data type concerned, the value at the affected output is set equal to this limit value.

Data types have been established for the call parameters and terminals of each PC element. For certain PC elements however, the data type for some terminals may be selected with a call parameter.

Many PC elements, for example time delays and counters, have different data types for control signal terminals and for data terminals.

The delay element in Figure 4-2 has data of type Boolean (B) at input I and output O. The input TD (Time Delay) and the output TE (Time Elapsed) on the other hand are of data type time (T).

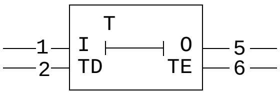


Figure 4-2. Delay Element TON

The counter in Figure 4-3 uses Boolean control signals and Integers (I or IL) for data input and output.

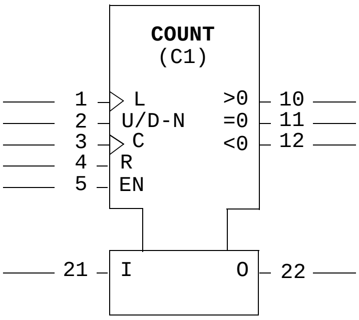


Figure 4-3. Counter COUNT

Each data type has a specified entry and presentation format. When data is entered via the Engineering Station, the format and value of the entered data is checked before it is converted to internal format.

Internal data is stored according to a defined format. As an example an Integer (I) is stored as a word of 2 bytes with range from -32768 to 32767.

4.1.1 Boolean

A Boolean (logical) variable can be either 1 or 0. The Boolean data type is used primarily for all logical PC elements but is also used for control signals, for example for data handling elements.

The AND-gate in Figure 4-4 has Boolean type data for all inputs and outputs.

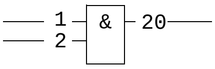


Figure 4-4. AND Function AND (2)

The multiplexer in Figure 4-5 uses Boolean values for control and for fault indication.

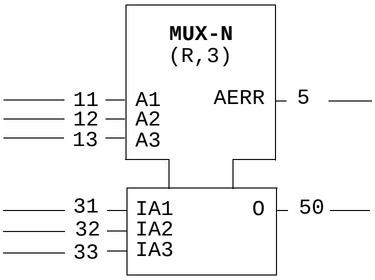


Figure 4-5. Multiplexer MUX-N (R,3)

Entry and Presentation Format

The entry format, that is the permissible values at program entry or test, are 1 and 0, 1 corresponding to true and 0 corresponding to false.

4.1.2 Integer

An integer (I or IL) can be positive, zero or negative. If no sign is specified the number is considered positive.

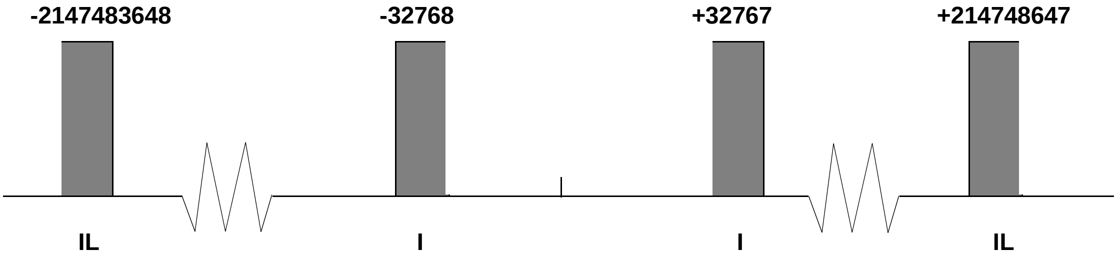


Figure 4-6. Number Range for Data Type Integer (I,IL)

There are two sizes of integers:

I	with the range -32768..32767 ($-2^{l5}..2^{l5}-1$).
	Internal storage form: 2 bytes,2’s complement
IL	with the range-2147483648..2147483647 ($-2^{3l}..2^{31}- 1$).
	Internal storage form: 4 bytes, 2’s complement.

NOTE

I and IL was previously designated I∗2 and I∗4 respectively These designations may still be used when entering programs.

The counter in Figure 4-7 is an example of the use of integers. The counter can be used with both types of integers. Which one is used in a particular instance is determined with a call parameter at the time the PC element is declared.

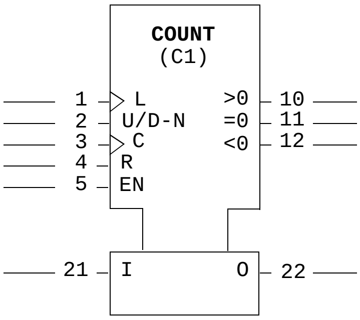


Figure 4-7. Counter COUNT

Another example of PC elements that can handle both integer types is the comparator COMP-I in Figure 4-8. The outputs of this PC element have Boolean values and the inputs have integer values, I or IL depending on the cal parameter C1.

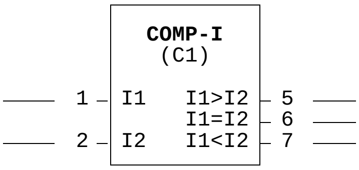


Figure 4-8. Comparator COMP-I for Integers

Entry and Presentation Format

When entering, the value and the number of digits is checked.

When entering integers a minus sign and digits may be used. After the last digit of the number a space or a carriage return < CR > must be entered. No spaces may occur between the minus sign and the digits or between adjacent digits.

Examples of integers:

1

4096

- 4711

-2

4.1.3 Real Numbers

This data type Real (R) is used primarily when processing continuous (analog) signals but also when very large or very small numbers must be handled.

Real numbers are sometimes also called floating point numbers. This refers to the internal representation, and thus will not be used here.

With the data type R both very large and very small numbers can be processed and stored. The range is approximately

$-9.2\ast10^{l8}$.. $-2.7\ast10^{-20}$

0.0

$5.4\ast10^{-20}$.. $9.2\ast lO^{l8}$.

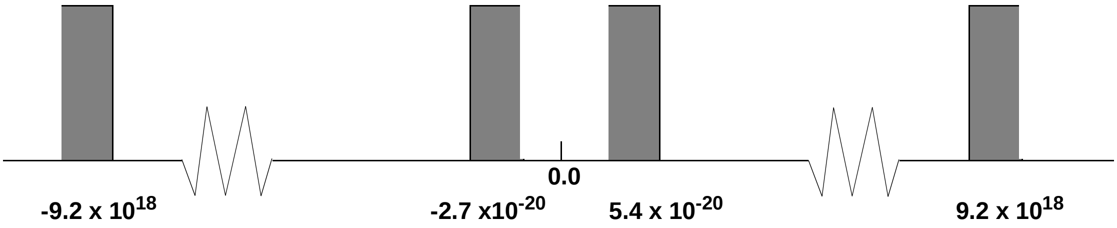


Figure 4-9. Number Range for Real Numbers (R)

Numbers between $-2.7\ast10^{-20}$ and $5.4\ast10^{-20}$ are given the value 0.0.

Internal storage format: 4 bytes floating point number according to IEEE format.

Most PC elements for arithmetic and data handling and PC elements for control and filter functions work with real numbers.

Figure 4-10 shows a PI regulator in which the outputs O and DEV and all inputs except TI, RINT and BAL are of type R.

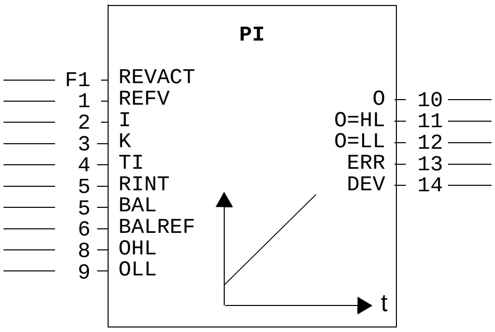


Figure 4-10. PI regulator PI

Entry and Presentation Format

Digits, “ + “, “-”, “.”, and “E” are permitted in a real number. A real number can be entered:

- as a number with or without decimal point, for example:

25770

12.459

-12.459

- in E (exponential) format with or without decimal point, for example:

the number 15750 may then be entered as

15.750E3

1.575E4

The number 1.235 may be entered as

1235E-3

The number 0.001 may be entered as

1E-3

An E must always be immediately preceded by a digit and also immediately followed by a digit or, if the exponent is negative, by the minus sign.

If a number contains a decimal point, this must either be preceded or followed by a digit.

Example:

0.25 must thus be entered as 0.25 or as .25

Spaces are not allowed in real numbers.

When generating PC diagrams, values that are larger than 10^6 or smaller than 10^{-6} are printed in exponential format.

4.1.4 Time

Two data types are available to represent time:

- T for times up to 24 hours.
- TR when times longer than 24 hours are required.

The data type T is for example used for delay elements and oscillators.

The data type TR (Time Real) is primarily used for control and filter functions.

For the delay element TON (Figure 4-11) the data type T is used for the Time Delay input and the Time Elapsed output

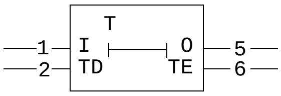


Figure 4-11. Delay Element TON

For the integrator INT in Figure 4-12 the type TR is used for the integration time, input TI.

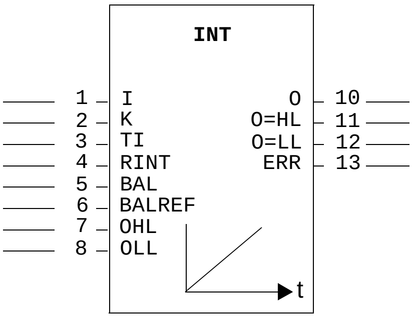


Figure 4-12. Example of Control Element

T	Time range: 0 ms - 23h 59m 59.999s
	Internal storage format: 4 bytes, positive integer
TR	Time range: 0 ms - $9.2*10^{l8}$ s
	Internal storage format: 4 bytes, positive real number

Entry and Presentation Format

A time value can be entered in the following format:

1. In milliseconds

   Given for the data type T as a positive integer followed by the unit designation ms.

   For example: 10 ms, 500 ms

   For the data type TR parts of a millisecond, for example 10.999 ms may be entered. This is, however, not meaningful since this precision cannot be maintained in calculations.

2. In seconds

   Given as a real number which may be followed by the unit designation s.

   The unit s is the default, that is if no unit is given s is assumed.

   For type TR exponential form (E-form) is also permissible.

   For example: 7.38,7.38s,5.2E2,5.2E2s

   E must always be directly followed by an exponent digit.

3. In minutes or hours

   Specified as a real number with unit designation m or h.

   For the data type TR exponential form can be used.

   For example: 25.3 m,18.25 h,1.25E2 h

4. In colon format

   The colon sign (:) is used to separate hours, minutes and seconds.
   Parts of second is given with a decimal point.

   For example: 6:25:30.2 = 6h 25m 30.2s

   25:30.2 = 25m 30.2s

   6:0:10 = 6h 10s

   A colon must be preceded by an hour or minute digit and followed by a minute or second digit respectively.

For both data types:

- A decimal point must be preceded by a digit and also be followed by one.
- A colon may not occur in times specified with a unit (h, m, s or ms).

4.1.5 Array

The data type Array (A) is primarily used for character strings. An array may contain up to 255 bytes.

For example in the element TEXT (Figure 4-13) the data type array is used at the output TLINE. The element converts different data types, for example integers (I or IL) or time (T or TR) into an array.

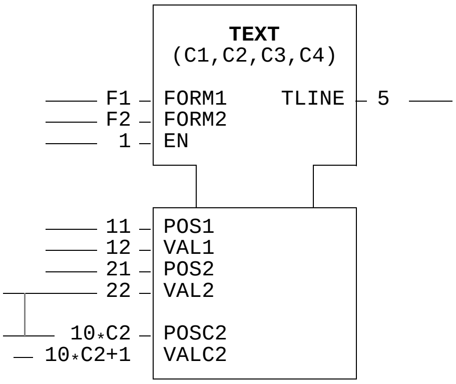


Figure 4-13. Text Element TEXT

The number of characters in the array may be determined by the element or be determined by a call parameter.

Entry and Presentation Format

A character string in ASCII format is entered as a sequence of characters enclosed in single quotes, for example:

“This is an Array (A) that contains 49 characters”

An array is internally stored as

- 2 bytes giving the length of the array (49 in the example)
- Data byte wise
- 1 byte for padding if the number of bytes is odd.

4.1.6 Group data

Group data (G) can be used to simplify the handling of a group of associated data of the same data type such as a group of 7 integers, or a group of 32 real numbers.

A group data unit is specified for a PC element in the same way as single data units.

Group data can be used, for example, to give input data to the PC element FUNG-1V, a function generator, see Figure 4-14.

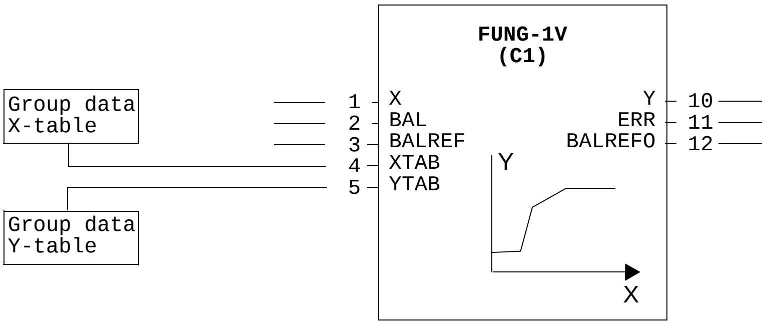


Figure 4-14. Group Data for the PC Element FUNG -1V (10)

An input signal X is to give an output signal Y which follows a specific function. The function generator gives a piecewise linear curve, by linear interpolation between break points.

A X value and an Y value is required for each break point. These values are stored as group data and are connected to the XTAB and YTAB inputs respectively.

The size of a group of data can be determined by the PC element itself or by a call parameter. In the example in Figure 4-14, the number is specified with “(10)”. (See further Section 4.4.1 Size Parameters.)

- The number of data in a group can be between 1 and 255.
- Any data type except Array may form a group.

Group data is stored as

- 2 bytes that specify the number of data in the group
- Number of bytes according to the space requirements for the type of the data in the group
- Padding to an even number of bytes.

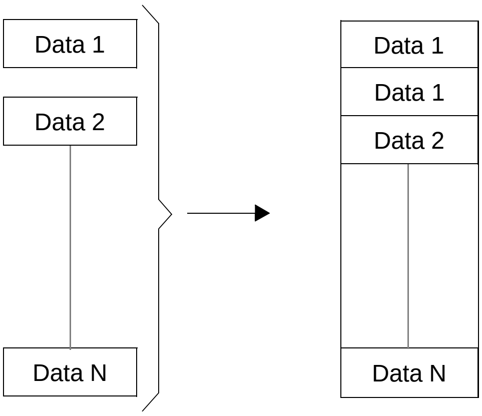


Figure 4-15. Group Data with N Data Units

Entry and Presentation Format

No special entry format exists for group data.

To form group data the PC element REG-G must be used, see Figure 4-16. This PC element compresses single data units at the inputs I1, I2, and so on to a group data unit at the output O. This output can then be connected to a group data input, see XTAB or YTAB in the example above.

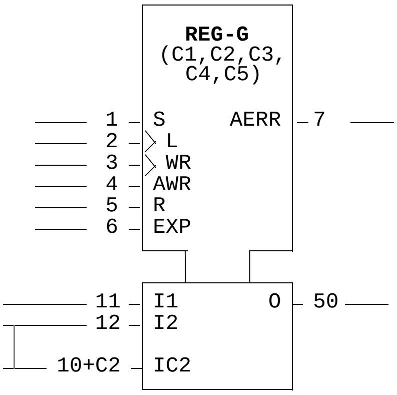


Figure 4-16. Group Data Register REG-G

4.2 PC Elements

The general graphic symbol for a PC element is shown in Figure 4-17.

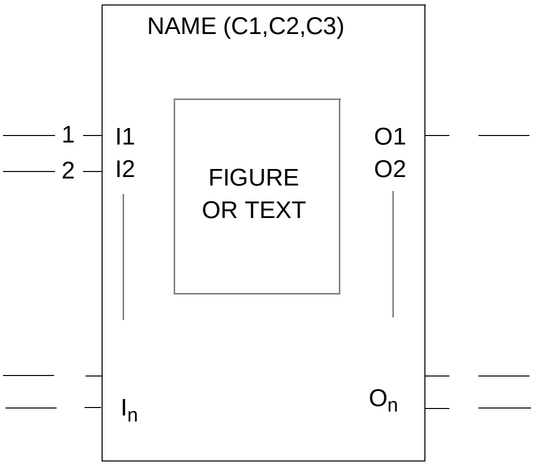


Figure 4-17. General PC Element Symbol

At the top of the symbol the call name and parameters of the PC element is usually given.

The symbol can otherwise be constructed in two ways:

- The PC element may contain a figure that illustrates the function of the PC element, as shown in Figure 4-18.
- The symbol is as far as possible drawn in accordance with available standard symbols or is in other ways self-explanatory.

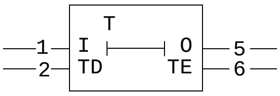


Figure 4-18. Example of Symbol with Function Symbol

For frequently used simple PC elements like AND, OR, ADD, and so on, the call name is replaced by a symbol and the call parameters are omitted. (See Figure 4-19.)

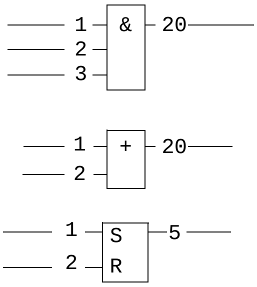


Figure 4-19. Example of PC Element Symbols

4.3 Call Names for PC Elements

The type of a PC element is specified with its call name and possibly one or more call parameters for PC element size, data type and so on.

The call name is unique and designed as a mnemonic abbreviation of the full PC element name. The name may consist of a maximum of 8 characters.

For example:

Counter COUNT

Switch SW

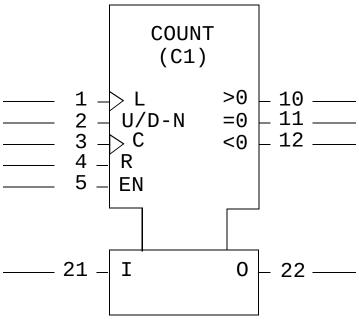


Figure 4-20. Counter COUNT

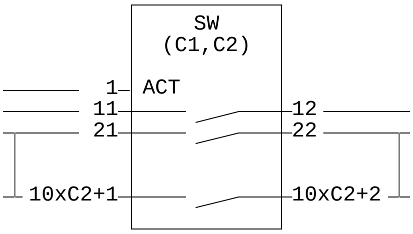


Figure 4-21. Switch PC Element SW

The call name may also contain an extra term, for example SR-D, REG-G or MUX-I. The auxiliary term indicates that the PC element has a special function or, as is the case for MUX-I, specifies the data type that is used for the address inputs.

SR-D indicates that the memory element SR has a data input (D input).

REG-G is a register element for group data and is used with, for example, function generators.

The conversion element CONV-IB is used for conversion from Integer to Boolean.

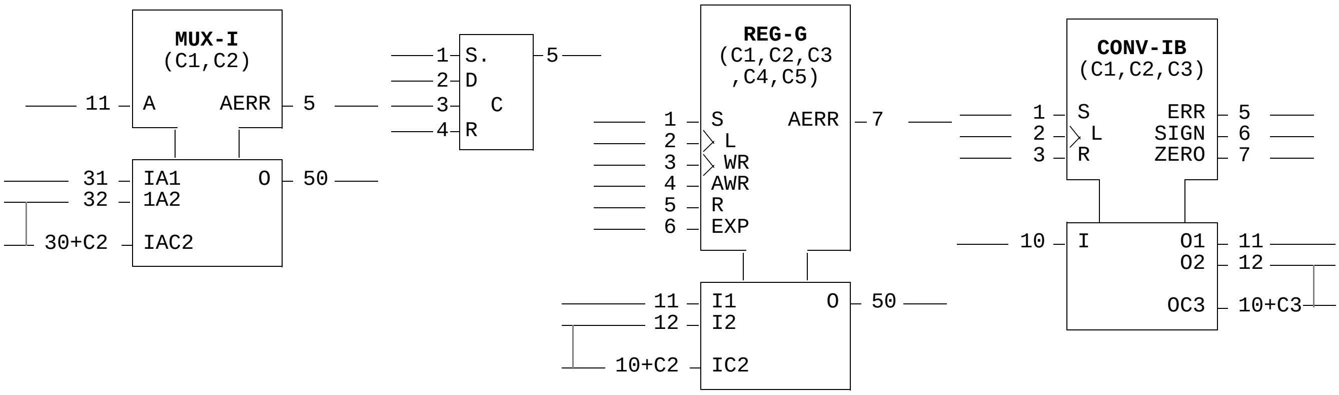


Figure 4-22. Example of Call Names with Auxiliary Designations

4.4 Call Parameters for PC Elements

Many PC elements are flexible with respect to size and data type. Others may have a fixed main function but different input and output configurations.

This information regarding size, data type and so on is specified with call parameters for entry in the programming tool and is stored in the PC program together with other information regarding the PC element.

4.4.1 Size Parameters

A size parameter may, for example, specify the number of inputs or outputs or number of groups of inputs or outputs. It may also specify the size of an internal storage area, for example, the number of positions in a shift register.

The calling sequence AND(C1) where C1 denotes the number of inputs is specified in the element data sheet for the AND function. The parameter must be an integer in the range 2..19, that is the smallest number of inputs is 2. A 3-input AND-gate is consequently called as AND(3).

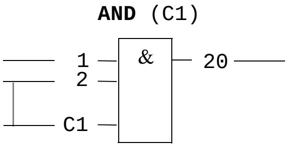


Figure 4-23. AND Element AND

A combined PC element of type AND-O may have up to 5 OR functions. The parameter C1 denotes the number of direct inputs to the AND element while C2.. C6 specify the number of inputs to the respective OR function.

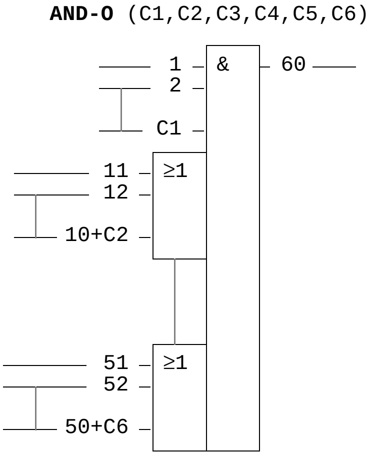


Figure 4-24. AND-OR Element AND-O

An AND-O element as shown in Figure 4-25 with one 3input and one 2-input Or function is called as AND-O (0, 3,2), where C1 = 0, C2 = 3 and C3 = 2. The parameters C4.. C6 need not be specified since they are 0. Note that C1 must be given although it is 0!. This is because the system on program entry always interprets the first value as referring to parameter C 1.

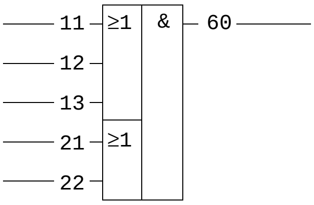


Figure 4-25. AND-OR Element AND-O (0,3,2)

A size parameter is used with the switch elements SW (Figure 4-26) and SW-C and with the fault signal element FAULT to specify the number of channels (groups).

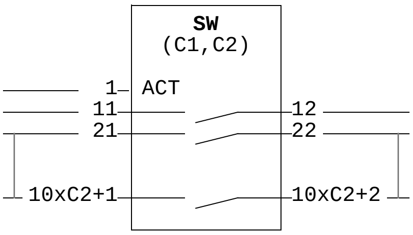


Figure 4-26. Switch Element SW

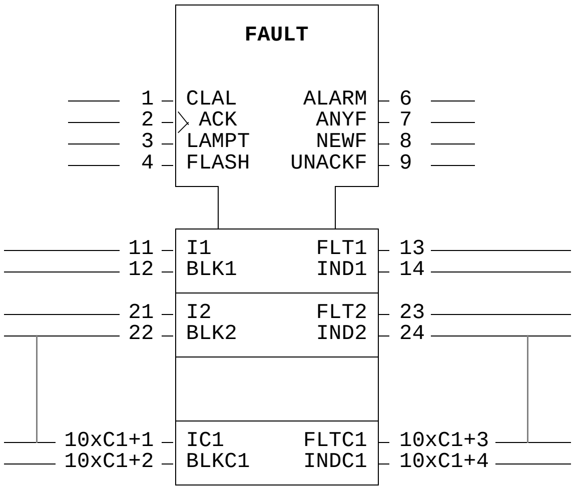


Figure 4-27. Fault Signal Element FAULT

A size parameter may also specify the size of an internal storage area. An example of this is the shift register SHIFT (Figure 4-28) where the parameter C2 specifies the number of places in the shift register.

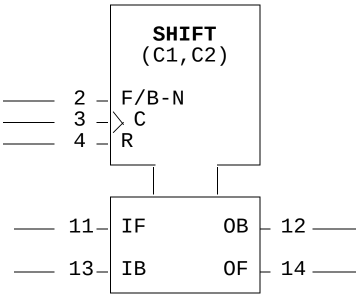


Figure 4-28. Shift Register SHIFT

In the queue register FIFO the parameter C2 specifies the number of parallel queues and the parameter C3 the number of places in each queue.

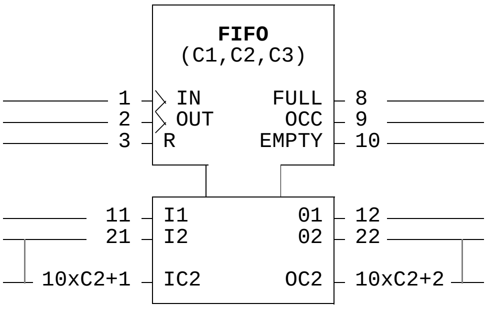


Figure 4-29. Queue Register FIFO

The size of the internal storage area does not influence the size of the symbol unless it also affects the number of inputs and/or the number of outputs.

4.4.2 Data Type Parameters

For PC elements that handle different types of data the type used in a particular instance of the PC element is determined with a data type parameter.

The data type is specified (in accordance with Section 4.1 Data Types) as

B	Boolean
I	Integer,16 bit
IL	Long integer,32 bit
R	Real number
T	Time
TR	Time specified as a real number
A	Array

NOTE

See Section 4.1.2 Integer regarding I∗2 and I∗4.

The data type for the addition element in Figure 4-30 is specified with call parameter C1 while C2 specifies the number of inputs. An ADD element for addition of 4 real numbers is thus called with ADD(R,4).

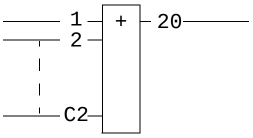


Figure 4-30. Addition Element ADD

A shift register for integers with 16 places is called with SHIFT(I,16)

The conversion element CONV is used to convert data of one type into another. The call parameter C1 determines the data type at the input and C2 at the output. To convert a long integer (IL) into a real number (R) this element is called with CONV(IL, R).

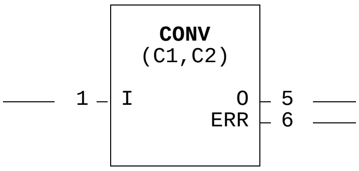


Figure 4-31. Conversion Element CONV

For most elements the chosen data type is given inside the symbol.

4.4.3 Variant-determining Call Parameters

Certain PC elements may have different variants of one and the same main function. The variant used in a particular case is determined with a call parameter given in the manner specified for PC element in question.

One example of this is the conversion elements CONVIB and CONV-BI that convert integers to Boolean and vice versa. The parameter C2 specifies the code that is to be used at the outputs (CONV-IB) respectively the inputs (CONV-BI).

C2 = 1 BC code

C2 = 3 BCD code

C2 = 3 1-of-N code

C2 = 4 Gray code

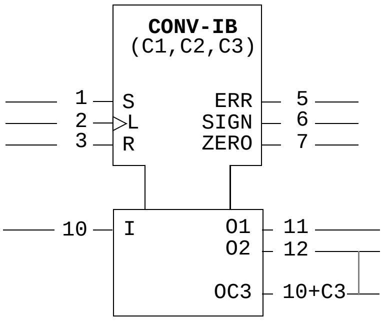


Figure 4-32. Conversion Element CONV-IB

4.4.4 Call parameters for structure elements

For the structure elements with execution control the following is given for the corresponding PC module using call parameters

- Cyclicity in ms.
- Order of execution among several PC modules having the same cyclicity.

See further Chapter 5 - Chapter 11.

4.5 PC Element Terminals

A PC element has a number of input and output terminals for connection to other PC elements, to the Data Base element, or to constants and operational parameters. Input terminals are always drawn on the left hand side of the symbol while output terminals are drawn on the right hand side.

4.5.1 Terminal Numbers

The terminals are identified with numbers in accordance with the PC element definition. The terminal number is specified outside the symbol frame, see Figure 4-33.

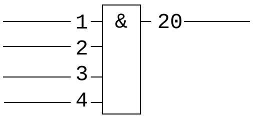


Figure 4-33. Terminal Numbers in PC Diagrams

An output terminal normally has a terminal number which starts on a multiple of 5, for example the number 20 for the output in Figure 4-33.

In PC elements with groups of input or output terminals, each group is normally allocated its own 10-series of terminal numbers. Examples of this are the memory elements SR-AA, -AO, -OA and -OO, see Figure 4-34.

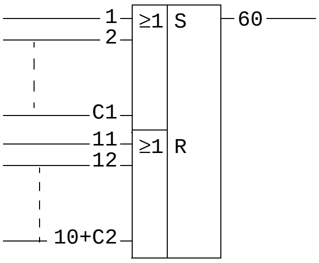


Figure 4-34. Memory Element SR-AO

If a PC element has groups containing both inputs and outputs, like the switch element inFigure 4-35, each such group is normally assigned its own 10-series. In tn example the activation input has the terminal number 1.

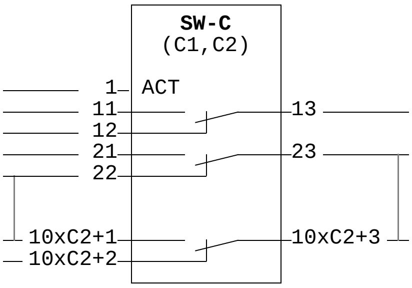


Figure 4-35. Switch Element SW-C

The terminal numbers are fixed for each PC element and cannot be changed by the user.

4.5.2 Terminal Names

A terminal often has an explanatory symbolic name. The name is written inside the symbol frame adjacent to each terminal. Figure 4-36 and Figure 4-37 show examples of this.

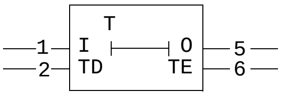


Figure 4-36. Delay Element TON

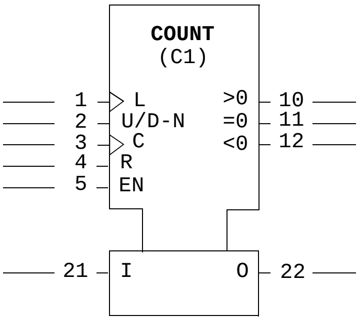


Figure 4-37. Counter COUNT

A data input to a PC element normally has the name I and an output the name O.

If several data inputs exist they are often numbered and thus get names I1, I2 and so on. The outputs get the names O1 02 and so on (see Figure 4-38).

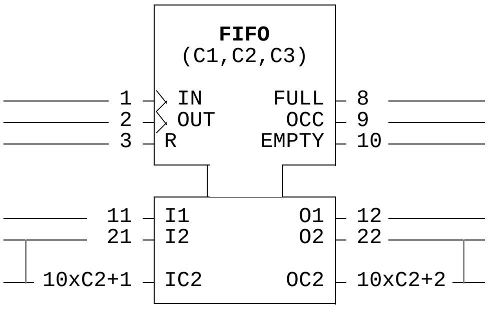


Figure 4-38. Queue Register FIFO

The terminal names are fixed for each PC element and cannot be changed by the user. They are given and explained in the PC element manual.

4.5.3 Terminal Type

In the PC element data sheet the data type of each terminal is stated. It may be given, for example, as IB for an Input of type Boolean, II for an Input of type Integer, and so on.

Examples of designations that may occur in the data sheet.

Table 4-1. Examples of Designations

For inputs	For outputs
IB	OB
II	OI
IIL	OIL
IR	OR
IT	OT
ITR	OTR
IGRC1	OGRC1

IGRC1 designates an input for group data. This group data contains C1 real numbers. OGRC1 designates an output for group data with C1 real numbers. C1 refers to a call parameter.

4.5.4 Dynamic Inputs and Inverted Signals

Figure 4-39 shows the symbol for a so called dynamic input. A dynamic input is active only when the input signal changes from 0 to 1 while a static input on the other hand is active while the input signal is 1.

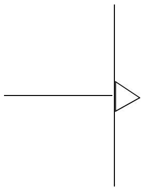


Figure 4-39. Symbol for Dynamic Input

An inversion of a logical input signal is always drawn outside the symbol, see Figure 4-40.

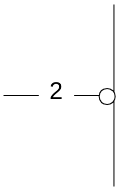


Figure 4-40. Inverted Input Signal

4.6 Symbol Types

In the introduction to Section 4.2 PC Elements, symbol layout was discussed on the basis of the general symbol form.

The basic form for a PC element symbol is a rectangle with fixed or variable size. The delay element TON in Figure 4-41 may serve as an example of a symbol with a fixed size.

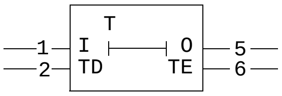


Figure 4-41. Example of Symbol with Fixed Size - TON

The AND function in Figure 4-42 is an example where the symbol size depends on the number of inputs.

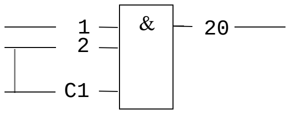


Figure 4-42. Example of Symbol with Variable Size - AND

An PC element may also be composed of several sub-elements, that is the PC element is constructed with several function levels.

Such a PC element may be fixed with respect to number of function levels and sub-elements but have a variable number of inputs.

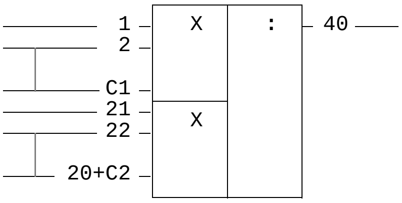


Figure 4-43. PC Element with Fixed Number of Sub-Elements but with Variable Size - DIV-MR

A combined PC element may be variable with respect to the number of sub-elements. The AND-OR element (ANDO) in Figure 4-44 may have up to 5 sub-elements, each with a maximum of 9 inputs.

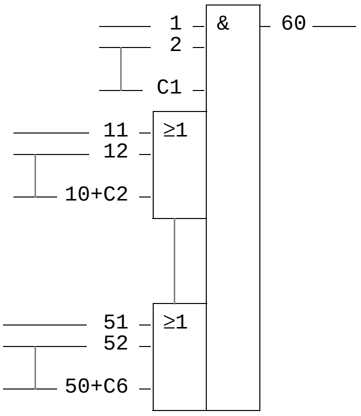


Figure 4-44. Combined Element - AND-O

A PC element may be composed of a common (superior) control part and a data section. Examples of such PC elements are (Figure 4-45) counters and registers. Both the control and the data part of these PC elements may be of variable size.

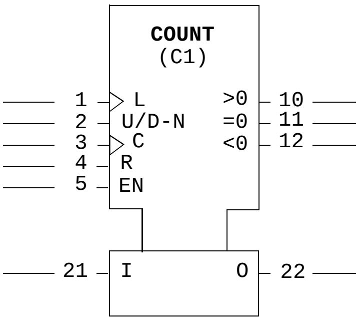


Figure 4-45. Counter COUNT

The switch elements SW and SW-C have, like the fault signal element FAULT, a size which varies with the number of contact groups (channels).

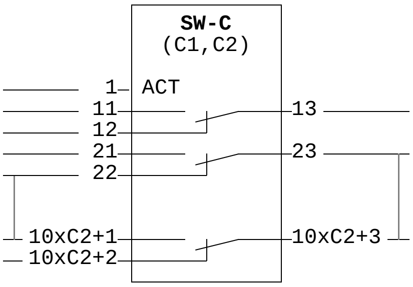


Figure 4-46. Switch Element SW-C

The symbol layout, terminal numbers, and so forth are fixed for each PC element and are shown in the *PC Element Advant Controller 400 Series Reference Manual*.

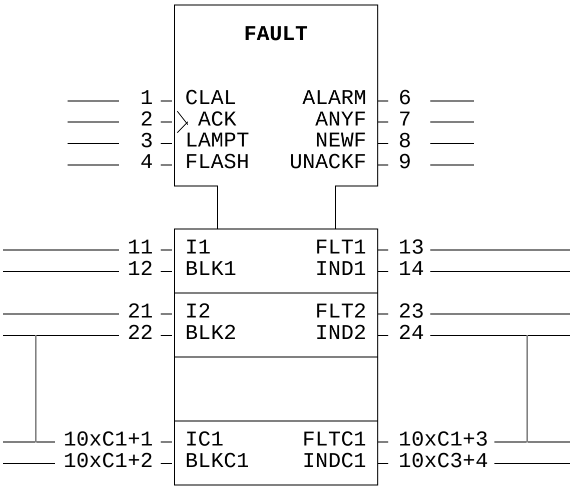


Figure 4-47. Fault Signal Element FAULT

4.7 Variables and Constants

Input data to an PC element may be

- Output variable from another PC element.
- Output variable from a Data Base element, for example representing the value from an input board.
- Operational parameter
- A constant

4.7.1 Output Variables

An output variable represents data at the output of a PC element. The counter COUNT in Figure 4-48 has four output variables, that each represents data calculated in the PC element during execution.

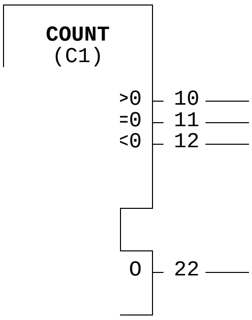


Figure 4-48. Output Variables - Counter COUNT

Storage space in the local data area for an output variable is automatically reserved at PC element definition time.

An output variable is normally identified with the identity of the PC element together with the terminal number and may also be given a symbolic name, see Section 4.8 Symbolic Naming of Variables and Constants.

The value of an output variable is transmitted to the input of a PC element by connection, represented by a connecting line in the PC diagram, see Figure 4-49.

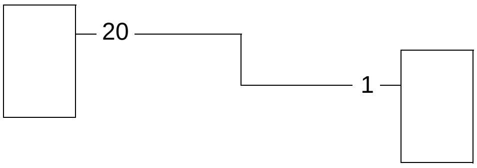


Figure 4-49. Connection Between Output Variable and PC Element Input in a PC Diagram

4.7.2 Connecting to Data Base Elements

When PC programming, data outside the PC program are referenced via Data Base elements. This reference is made by a simple notation in the PC diagram and may be directly handled by the Engineering Station. This is illustrated in Figure 4-50.

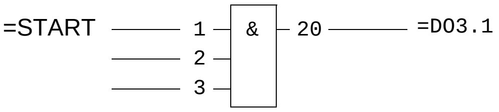


Figure 4-50. Connection to Data Base Elements

The equal-sign (=) denotes that the connection is to a Data Base element.

Either the name or the board channel designation may be used for identification.

When the terminal VALUE of the Data Base element is referenced this name need not be given.

Example:

= START

When referencing other terminals of the Data Base element the terminal number (property number) or name must be given preceded by a colon (:).

Example:

=DI3.2:17

=START:ERR

It i s also possible to use purely numerical identities. These are composed of file and record number as shown enclosed in parentheses in the Data Base element header followed by the terminal number.

Example:

=6.1:23

4.7.3 Operational Parameters

An operational parameter is a local variable (within the PC program) which normally has a fixed value during execution. An operational parameter can be changed, even during execution, from a service tool.An operational parameter may, for example be used to specify the delay time for a delay element, see Figure 4-51. The operational parameter is specified with D = < value>. In the example the delay time is 10 secs.

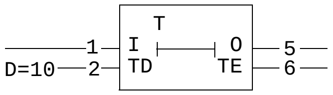


Figure 4-51. Example of Specification of Operational Parameter

4.7.4 Constants

A constant is, like an operational parameter, a local variable whose value remains fixed during execution. A constant is stored in such a way that it is only accessible for changing with a programming command, thus differing from an operational parameter.

A constant is specified with CD = < value >, see Figure 4-52.

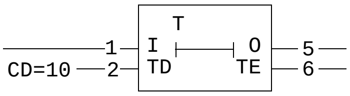


Figure 4-52. Delay Element TON

Constants of Boolean type, that is with the values 1 and 0, have been predefined in the PC system. For these constants it is not necessary to specify CD = < value >, but it suffices to directly write 1 or 0, as shown in Figure 4-53.

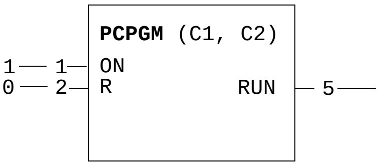


Figure 4-53. Example of Using Boolean Constants

4.7.5 Multiple Use of Operational Parameters and Constants

Each user-defined operational parameter and constant is stored in a unique location within the local data area of the PC program. An operational parameter or constant may be common to several PC elements, for example as a common value for several time delays or counters.

Such common operational parameters or constants are in the documentation designated as multiple-use data specified with MD = or MCD = .

Figure 4-54 shows an example of use of multiple data. Both delay elements have been given the same delay time.

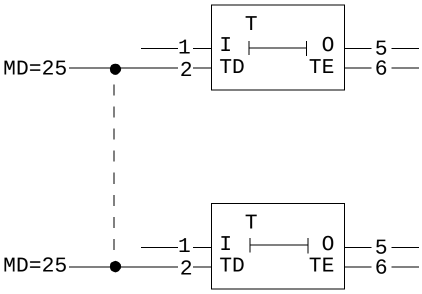


Figure 4-54. Multiple-use of Operational Parameter

A multiple-use operational parameter is changed in the same way as a normal operational parameter with a command from an Engineering Station. A multiple use constant can only be entered and changed by a programming command.

Data is referred to with MD respectively MCD or with D respectively CD.

Following a change the documentation is automatically changed so that the necessary M is inserted if the parameter or constant is multiply used.

4.7.6 Function Parameters

Function parameters are used when the value at a terminal is to be constant for a particular application. The input REVACT for the PC element PI in Figure 4-55 is an example of such a terminal.

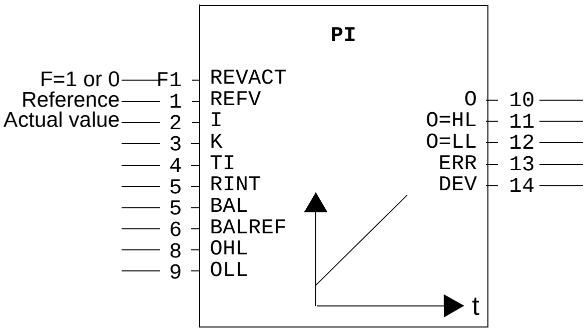


Figure 4-55. <PI Function PI

The function parameter REVACT (Reverse Action) determines if the output signal is to increase or decrease with increasing actual value.

Function parameters are identified with a number preceded by an F (F1 in Figure 4-55).

The value for a function parameter is given directly in front of the terminal and is entered during the so called connection command. Changing the value causes an initialisation of the PC Element.

Which inputs are function parameters is determined by the PC element and the user cannot influence this.

The value is specified in the draft diagram by the terminal thus:

F = < value >. The value is entered according to the format for the data type of the terminal.

4.8 Symbolic Naming of Variables and Constants

Variables and constants may be given symbolic names. The name is given in the PC diagram.

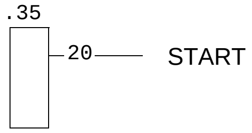


Figure 4-56. Symbolic Name of Output Variable

A symbolic name:

- Must be unique within a PC program.
- Must begin with a letter.
- May be up to 20 characters long.
- May contain letters, digits and underscores (_). If the name is enclosed in parentheses special characters may be used. The parentheses are counted as part of the name, and the name is printed with them.
- Upper and lower case letters may be used.

The system does not differentiate between lower case and upper case letters and the name “TID1” is thus considered the same as “tID1”. When printing the PC diagram the name is printed the same way it was entered at the definition.

4.8.1 Output Variables

An output variable may be identified by the item designation of its terminal. The output variable may also be given a symbolic name which may be used as an alternative when connecting PC elements.

4.8.2 Operational Parameters and Constants

Operational parameters and constants may be given symbolic names, as can multiple-use operational parameters and constants.

Figure 4-57 shows an example of the use of a symbolic name for a multiple-use operational parameter. The name is defined when entering the parameter into the Engineering Station. After defining the name the parameter may be connected to other PC elements by using the name.

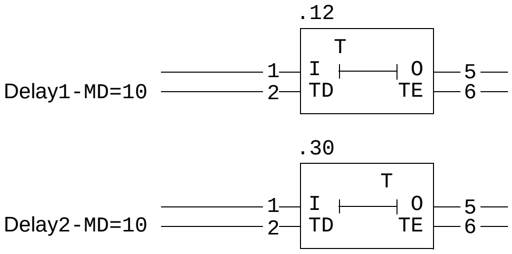


Figure 4-57. Example of Symbolic Naming of Operational Parameter

Chapter 5 Structure Elements of the PC Language

The structure PC elements in the Advant Controller 400 Series system are a group of superior PC elements with two purposes:

- Using structure PC elements the PC program may be arranged into hierarchically ordered function sections, modules, in accordance with the structure of the process.
- The structure PC elements control, on a module by module basis, the execution of the PC elements in the program.

Each function section (PC module) has a structure PC element, a so called module header. In the PC diagram the module header is placed in the upper left corner of the module, se Figure 5-1.

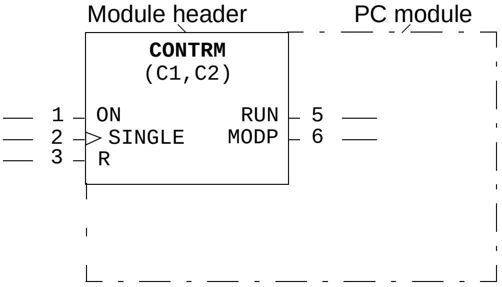


Figure 5-1. Module Header in a PC Module

5.1 Dividing the PC Program into Functional Sections

To obtain an easily handled control system the program should be divided into functional sections. The partitioning of the program is usually adapted to the functional structure of the plant, process or machine that is to be controlled. The need may arise to turn on or off different parts of the program at different times. Different parts of the process may have different timing requirements and thus different program modules may need to be executed with different cycle times.

A well structured program gives an easily over viewed documentation and promotes efficient program development, commissioning, and maintenance.

An example was given in Chapter 2 General of a pump station with two pump units with common control. The structure of the installation has the following appearance:

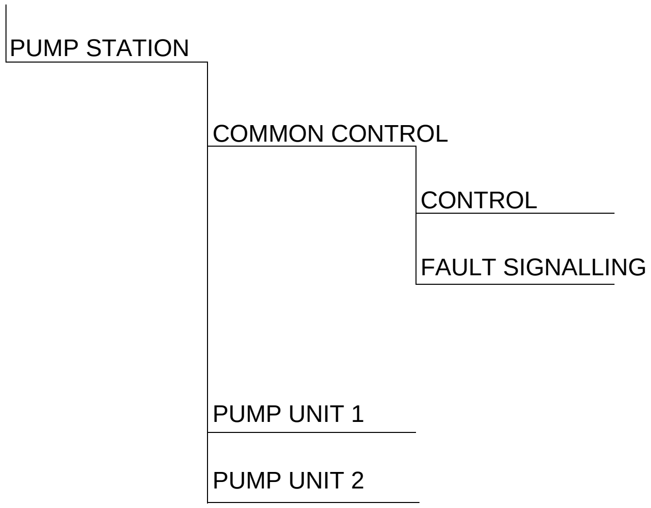


Figure 5-2. Pump Station - Structure of the Installation

The corresponding subdivision of the program into PC modules then becomes:

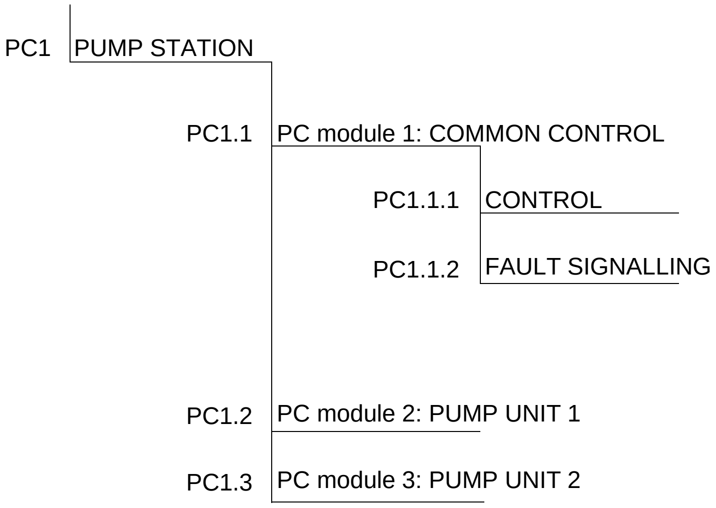


Figure 5-3. Pump Station - Structure of the PC Program

The item designations of the PC modules are in accordance with the principles specified in Section 2.4 Program Modules.

The common co-ordinated control is thus allocated a PC module with designation PC1.1 and the individual control modules are designated PC1.2 and PC1.3. (Refer to Figure 2-5 in Section 2.3 PC Programming.)

Different parts of a control program may need to be connected at different times. As an example, the control of a machine which operates in two modes.

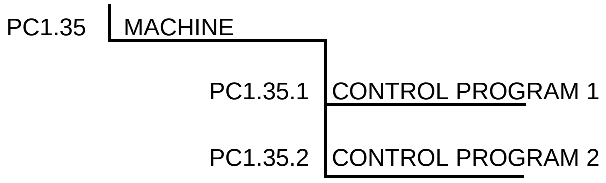


Figure 5-4. Machine Control with Alternative Programs

Control Program 1 may be an automatic program while Control Program 2 is a program for manual operation.

An example in which the division of a program into modules is determined by the need for different cycle times for different parts of the program is shown in Figure 5-5.

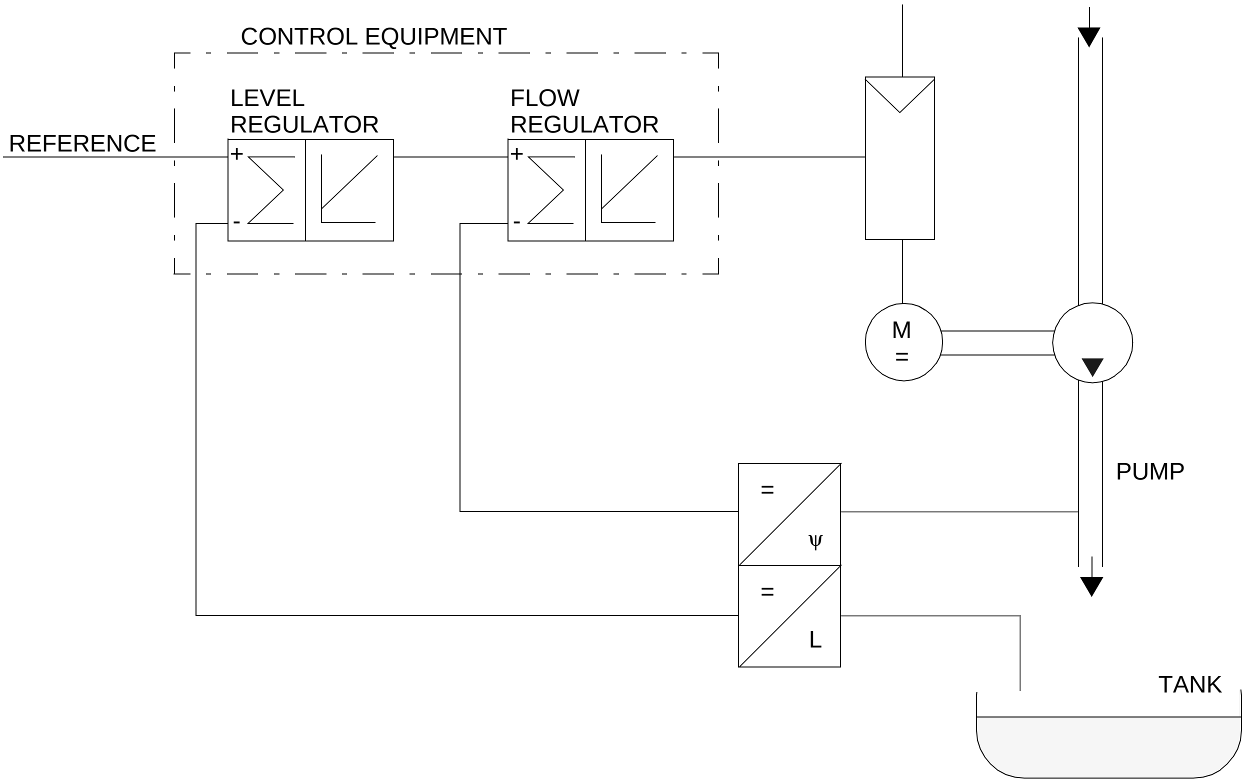


Figure 5-5. Level Controller with Speed-Controlled Pump

This example illustrates a control system for a vessel which is to be filled to a preset level via a pipeline using a speed-controlled pump. Such a system may contain a slow level controller and a fast flow controller which in turn gives a speed reference to the pump drive.

To permit the use of different cycle times, the controllers are included in separate PC modules located on the same level in the program structure. (This is discussed later in the text.)

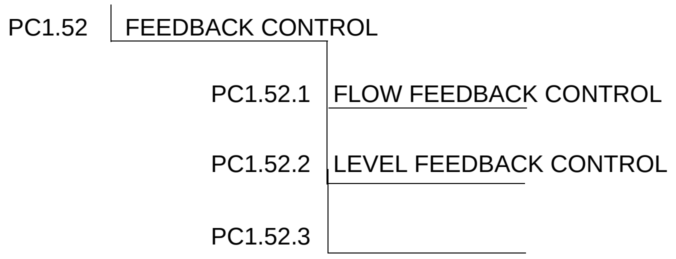


Figure 5-6. PC Modules- Different Cycle Times

It is sometimes desirable to switch the execution of a part of a PC program on or off. As an example refer to the dual pumps of Figure 2-5. Maintenance may require one of the pumps to be disconnected for a period of time. To enable this the control for the individual pump units can be put in different PC modules.

The subdivision of a PC program into function sections should not be carried further than that each function section corresponds to a clearly limited function. A subdivision into too many sections may cause confusion.

The system permits subdivision into 9 levels.

Rules for Program Subdivision into Function Sections

1. Basic rule. The subdivision of the program should parallel the division of the plant, process or machine.
2. Give each independent plant section its own PC program, the highest level in the program structure.
3. If the execution of any part(s) of the program must be different from that of other parts, the program should be further subdivided, if necessary, to allow for this.
4. Attempt to develop standardized program modules which may be reused in other programs.

5.2 Execution Units

The PC elements in a PC program are executed cyclically and module by module. Execution controlling modules are selected for the different function sections to control their execution.

Every execution unit is given a cyclicity, that is a time between executions of the unit. The PC elements within such a unit are executed in the order they were entered at program entry.

The execution units have special inputs for connection of conditions to switch execution of the unit on or off.

It is also possible to switch execution of the entire PC program on or off.

The generally usable execution unit is called control module. Its module header is called control module header (CONTRM), refer to Chapter 7 Control Modules - CONTRM.

Sequence modules with sequence headers (SEQ) are provided for sequential control.

The sequence header is a superior unit for a number of step modules that is included in a sequence. See Chapter 10 Sequence and Step Modules.

Another type of execution unit is the master-slave combination. The master header is the superior part of this execution unit for a number of slave modules. The slave units are used to collect different parts of the program into one execution unit.

The master-slave combination is described in Chapter 9 Master-Slave Combination.

Each PC program must have a PC program header (PCPGM). The task of the program header is to provide superior control of the execution of all execution units in the program. The program header for Advant Controller 400 Series is described inChapter 8.

When the functional subdivision of the program into function sections has been done, it remains to select the proper execution unit for each section.

If several adjacent function sections in the program structure may be assigned the same execution conditions and may be executed with the same cyclicity, these sections can be combined into the same control module. To retain the program structure each of the function sections are maintained as function modules. An example of this is given in the co-ordinated control of the pump station, see Figure 5-7.

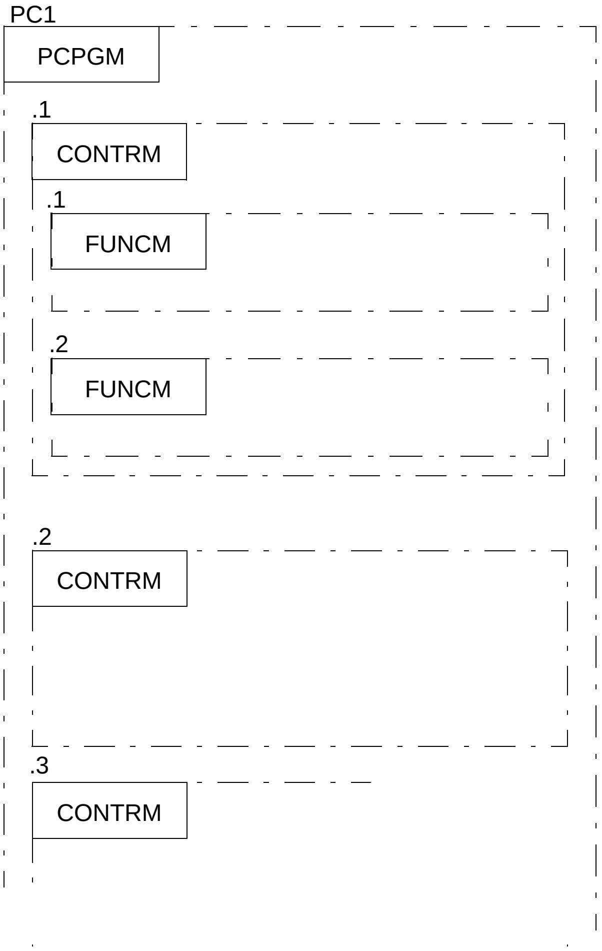


Figure 5-7. Module Structure for Pump Control

Another structure is illustrated by the level controller in Figure 5-5. The two controller functions (with different cyclicity requirements) are programmed as control modules. These control modules are then combined into one function module (Figure 5-8) designated “CONTROLLER -LIC105” with item designation PC-.52.

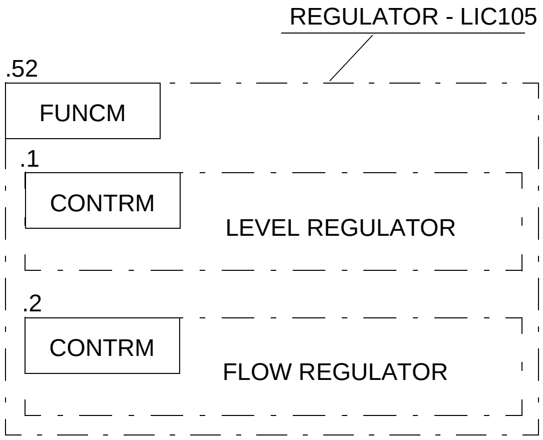


Figure 5-8. Module Structure for Level Controller

The function module has only a documentation purpose, that is it makes it possible to retain the desired program structure.

Chapter 6 through Chapter 10 describes the different execution units of the Advant Controller 400 Series system and the rules that apply to their use.

Detailed descriptions of the functions of the structure PC elements can be found in the *PC Elements Advant Controller 400 Series Reference Manual*.

5.3 Hierarchy of the Module Headers

In a PC module can be included either other modules or function PC elements, that is function PC elements and module headers may not be mixed within a module. The exception to this rule is PC-blocks (BLOCK), that may be mixed with function PC elements.

A module may thus be composed in one of the following ways.

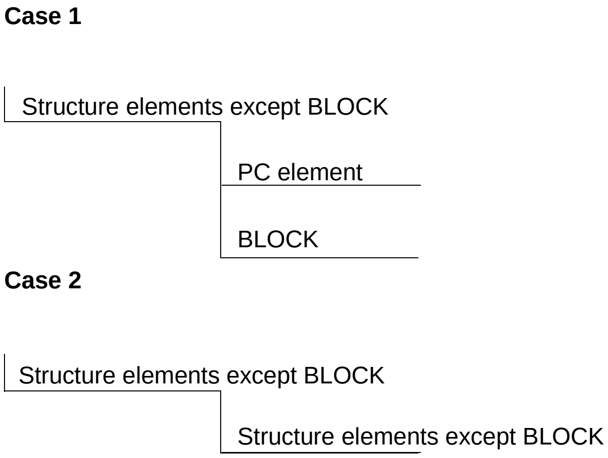

Case 1
Structure elements except BLOCK
PC element
BLOCK
Case 2
Structure elements except BLOCK
Structure elements except BLOCK

Figure 5-9. Module Composing

The subordinate modules in case 2 may then in turn be composed according to one of the 2 cases. Specific rules for each header is given in the appropriate chapter.

Chapter 6 Function Modules- FUNCM

6.1 General

A function module is created by introducing a function module header, see Figure 6-1, into the program hierarchy.

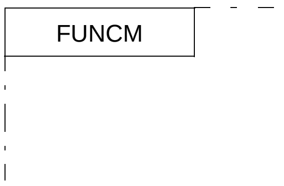


Figure 6-1. Control Module Header for a Function Module

A function module has its own identity in the program hierarchy.

The PC elements in the function module are given identities in the corresponding way.

A function module has no execution conditions, and thus it must be part of an execution unit, or alternatively contain execution units.

A function module may itself contain function modules but, at some level of the hierarchy there must exist a control module, a slave module or a sequence module to determine execution for the PC elements of the program. Refer to the chapters on these modules.

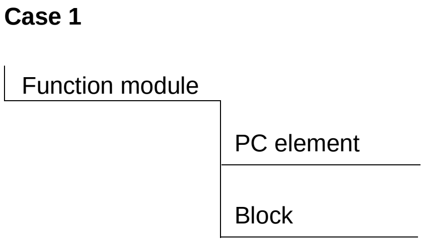


Figure 6-2. Subdivision of a Function Module - Case 1

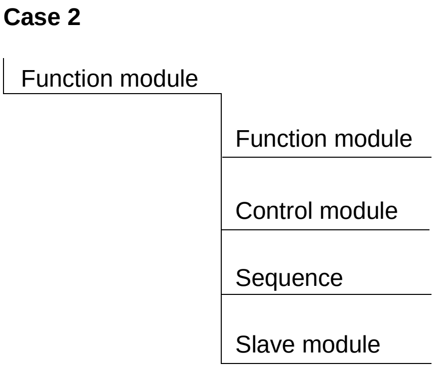


Figure 6-3. Subdivision of a Function Module - Case 2

Chapter 7 Control Modules - CONTRM

The control module is the most general execution unit in the Advant Controller 400 Series system.

The PC elements in a control unit are always executed in documentation order and with the cyclicity specified for the module.

The control module header is the PC element that controls the execution of the module in accordance with the conditions in effect. Its symbol is shown in Figure 7-1.

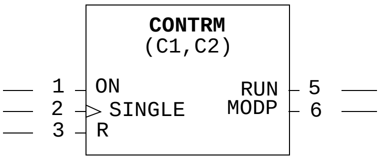


Figure 7-1. Control Module Header CONTRM

On/off control of the execution of the module is performed using the control inputs ON and R (Reset). Single-cycle execution is started with the input SINGLE. These control inputs are normal PC element inputs and may be connected to variables, operational parameters or constants.

7.1 Call

The control module header is called with CONTRM(C1, C2), see Section 4.3 Call Names for PC Elements and Section 4.4 Call Parameters for PC Elements. The call parameters have the following significance:

C1	The required cyclicity for the module in ms. The cyclicity may be chosen from those available for the Advant Controller 400 Series system used.
C2	The sequence of execution when more than one execution unit has the same cyclicity. If no value is specified the programming tool selects the first free slot in the so called cycle time table for the selected cyclicity. (See Chapter 14 Execution.)

The parameter C1 must always be given. If the Engineering Station is to select execution order the control module header is called with CONTRM(C1).

7.2 Control

The execution of a control module is controlled with the control inputs ON, R and SINGLE. Figure 7-2 shows a function diagram for the function of the control module.

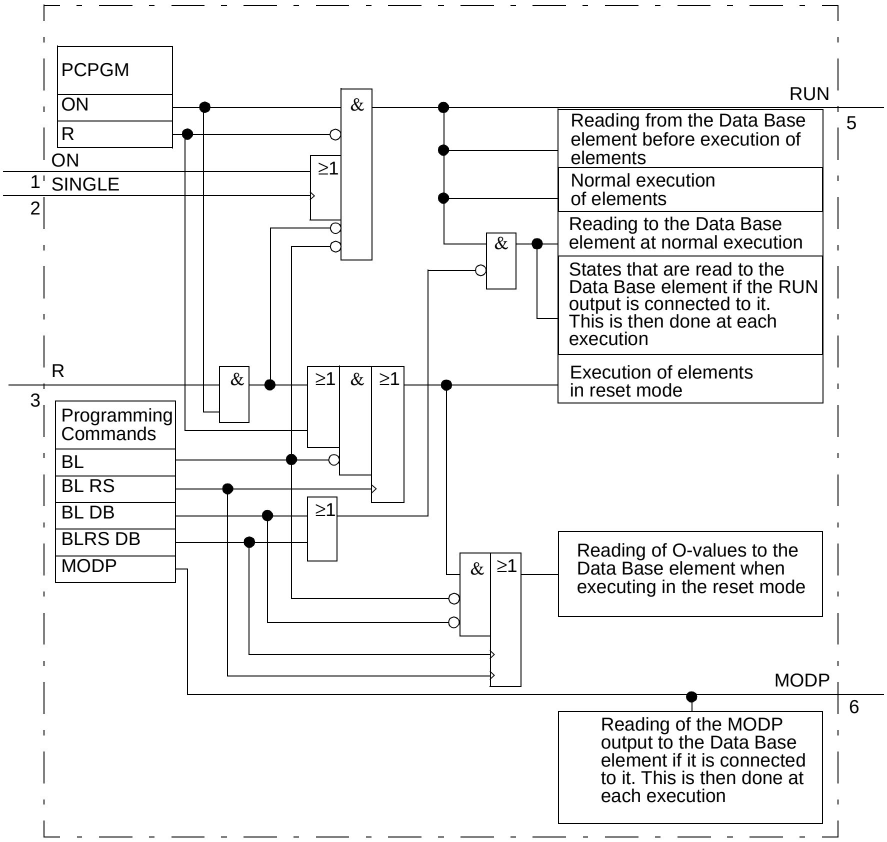

PCPGM
ON
R
ON
1
SINGLE
2
R
3
Programming Commands
BL
BL RS
BL DB
BLRS DB
MODP
&
≥1
&
≥1
&
≥1
&
≥1
& ≥1
RUN
5
Reading from the Data Base element before execution of elements
Normal execution of elements
Reading to the Data Base element at normal execution
States that are read to the Data Base element if the RUN output is connected to it. This is then done at each execution
Execution of elements in reset mode
Reading of O-values to the Data Base element when executing in the reset mode
MODP
6
Reading of the MODP output to the Data Base element if it is connected to it. This is then done at each execution

Figure 7-2. Function Diagram for CONTRM

A control module may be executed in two different modes normal mode and reset mode.

In the normal mode all PC elements are executed with the current input data to the module and calculated data is output to the Data Base element or to output PC elements for use in other execution units.

In the reset mode (clearing) all PC elements are reset to their initial states. This normally implies that the value 0 (for each data type) is output to Data Base elements and to the output PC elements. This mode is obtained if the R input is active.

The internal program system gives a start signal for execution with the cyclicity given by the parameter C1.

A more complete description of how execution is performed in the Advant Controller 400 Series system is found in Chapter 14.

ON = 1

If ON is 1 and R is 0, the module is executed in normal mode in accordance with the parameters C1 and C2.

SINGLE = 0--> 1

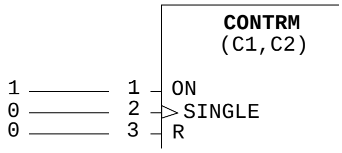


Figure 7-3. Control Inputs to the Control Module Header

If both ON and R is 0 the module may be executed once in normal mode according to C1 and C2 by setting the input SINGLE to 1.

The SINGLE input may also be inverted, in which case the execution starts when SINGLE goes to 0.

NOTE

Inputs to execution units can not be inverted if they are connected to the Data Base element.

R = 1

If R is 1 the module is executed in reset mode according to the parameters C1 and C2.

The R input overrides the ON and SINGLE inputs.

If all control inputs are 0 the module is not executed. This implies that all output data remains as they were following the last execution. No read-outs are performed.

7.3 RUN-indication

As can be seen in the function diagram in Figure 7-2, the RUN output is set to 1 if the module is executed in normal mode.

With SINGLE execution, RUN is 1 only during the one cycle.

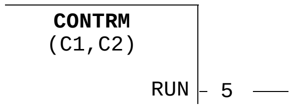


Figure 7-4. RUN-Output from the Control Module Header

7.4 Blocking from the Engineering Station

The execution of a control module can in different ways be blocked from an Engineering Station.

Table 7-1 shows the functions that are available to block one or more modules. Each new blocking command is performed independently of previous blockings or actual execution status. More information is available in the manual for the Engineering Station used.

Table 7-1. Blocking functions from the Engineering Station

Action	Effect on Module Specified
Block	Blocks the modules. Freezes output variables
Block output to the Data Base element	Only output to the Data Base element is blocked
Reset and block	Resets the modules and blocks output to Data Base element
Output to Data Base element	Blocks output to the Data Base element
Cancel blocking	Blocking released
List blocked execution units	Lists those modules that are blocked or in build mode

7.5 Modify Permission

The module headers for control, slave, and sequence modules contains a built-in check for the permission to change - Modify Permission.

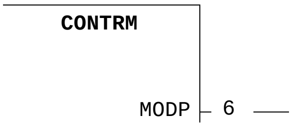


Figure 7-5. MODP Output from the Control Module Header

If the modify permission is deactivated, that is MODP is 0, all attempts to change the function of a module from a service tool are rejected. In this state only operational parameters may be changed.

Modify Permission can be activated and deactivated from the Engineering Station, with special commands. A list of the state of Modify Permission for selected modules can also be obtained from the tool.

In the initial state, that is when a module is created, Modify Permission is activated. By deactivating Modify Permission, for example when commissioning, it is indicated that for example the module has been completely tested.

7.6 Execution of a Control Module

The PC elements in a control module are always executed cyclically in the order they were entered from the Engineering Station. This order also applies to the documentation.

An execution starts with reading the input variables used in the module to the local data area. Following this the PC elements in the module are executed and finally, the output variables are output to the Data Base element.

7.7 Control Module in the Hierarchy

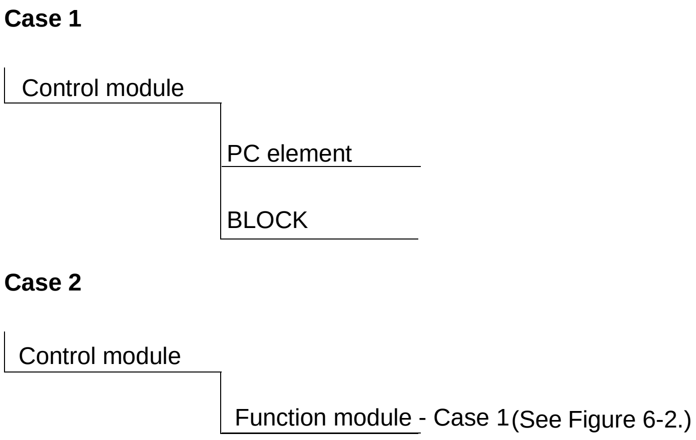


Figure 7-6. Control Module in the Hierarchy

A control module

- May be included together with other control modules in a function module
- May contain a number of function modules
- May contain both PC blocks and PC elements

Chapter 8 PC Programs in Advant Controller 400 Series

8.1 General

A single Advant Controller 400 Series system may contain several PC programs. Each program normally corresponds to a plant or process section. The maximum number of PC programs is defined at configuration of the system. These PC programs are then identified with a number from 1 to the specified maximum.

A PC program must contain one or more execution units such as control modules, sequences or master-slave combinations to have a meaningful function.

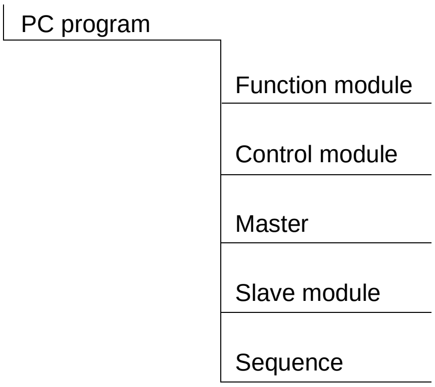


Figure 8-1. PC Program in the Hierarchy

The PC program is a unifying frame for a number of execution units. It is also possible to turn execution of all execution units contained in the program on and off from the program control header.

The symbol of the program control header is shown in Figure 8-2.

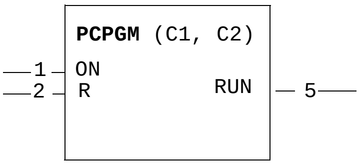


Figure 8-2. Control Header for PC Program

The control header of the PC program is given its own cyclicity. When it is executed the control inputs ON and R are tested when the header is executed. These inputs affect the execution of the entire program.

8.2 Call

The PC program is called with PCPGM(C1, C2), see Section 4.3 Call Names for PC Elements and Section 4.4 Call Parameters for PC Elements. The call parameters have the following significance:

C1 The required cyclicity for the PC program header in milliseconds. The cyclicity must be chosen from those available for interpreter A in the Advant Controller 400 Series system used.

C2 The sequence of execution when more than one execution unit has the same cyclicity. If no value is specified the programming tool selects the first free slot in the so called cycle time table for the selected cyclicity. (See Chapter 14 Execution.)

The parameter C1 must always be given. If the Engineering Station is to select execution order the control module header is called with PCPGM(C1).

8.3 Control

ON = 1

If ON is 1 and R is 0, the entire PC program is executed in accordance with the conditions that apply to the individual execution units.

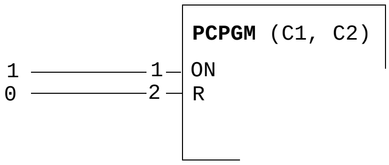


Figure 8-3. Control Inputs for Program Header

R = 1

If R is 1 all subordinate execution unit are executed in reset mode regardless of the execution conditions for these and regardless of the value at input ON.

ON = 0, R = 0

If all control inputs are 0 the execution of all subordinate execution units is blocked.

If the PC program header is activated, that is ON = 1 and R = 0, the RUN output is 1.

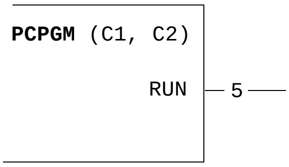


Figure 8-4. RUN Output from Program Header

8.4 Blocking from the Engineering Station

A PC program may, in the same way as a control module be blocked from the Engineering Station.

The blocking of a PC program has the effect of blocking each execution unit individually. A blocking entered for the entire PC program may then be released execution unit by execution unit. By deblocking at the PC program level all blockings introduced for individual execution units will be removed at the next execution time for each module.

Chapter 9 Master-Slave Combination

9.1 General

Master-Slave is used when it is desired to combine modules from different parts of the program hierarchy to a single execution unit keeping the functional structure unchanged. A master-slave combination is comparable to a control module divided into several function modules.

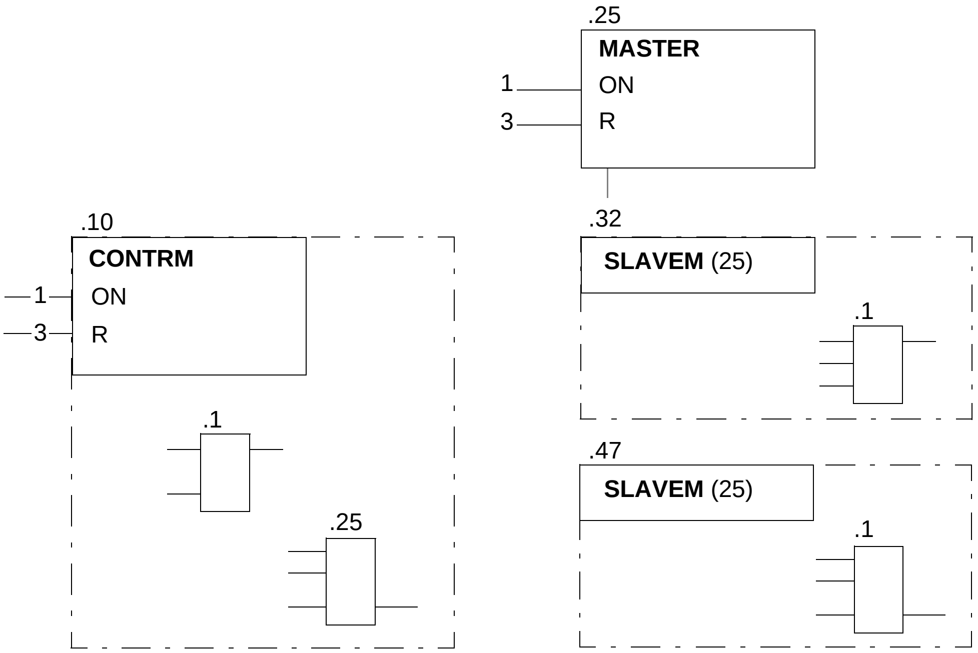


Figure 9-1. Comparison Between Control Module and Master-Slave Module

The master header thus corresponds to the control module header and controls the execution of those PC elements that are included in the slave modules which belong to the master header. Several master-slave groups may be included in a PC program. The individual slave modules may be located at different levels in the hierarchy. They may also be included in function modules together with control modules or other slave modules.

The master-slave group is executed as a unit, that is the slave modules belonging to one master are all executed after each other in the order of documentation.

9.2 Master Header - MASTER

The master header is given its own identity that must be at the highest level directly under the PC program.

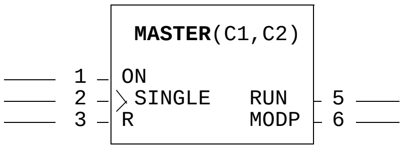


Figure 9-2. Master Header MASTER

9.2.1 Call

The master header is called with MASTER (C1, C2), see Section 4.3 Call Names for PC Elements and Section 4.4 Call Parameters for PC Elements. The call parameters have the following significance.

C1 The cyclicity for the master-slave group in milliseconds. The cyclicity must be chosen from those available for the actual Advant Controller 400 Series system.

C2 The order of execution when more than one execution unit has the same cyclicity. If no value is specified the Engineering Station selects the first free slot in the so called cycle time table for the selected cyclicity (see Chapter 14).

The parameter C1 must always be given. If the Engineering Station is to select execution order the master header is called with MASTER(C1).

9.2.2 Control

The master header has the same control inputs as the control module header.

If ON = 1 and R = 0 the master-slave group is executed in normal mode according to the parameters C1 and C2.

With single cycle execution (SINGLE = 0 -- > 1, ON = 0

If R = 1 the entire master-slave group is executed in reset mode irrespective of the values at inputs ON and SINGLE.

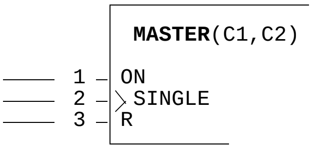


Figure 9-3. Control Inputs for Master Header

9.3 Slave Module - SLAVEM

Figure 9-4 shows the symbol of the slave module header.

Figure 9-4. Slave Header SLAVEM

9.3.1 Call

The slave module header is called with SLAVEM(C1).

The call parameter C1 specifies the master header to which the slave module is to belong. The value of C1 must be the last part of the item designation of the master header.

In the example in Figure 9-5 the master header has the item designation PC1.25. For all slave modules that are to belong to this master the parameter C1 is given as 25, the last part of the item designation of the master.

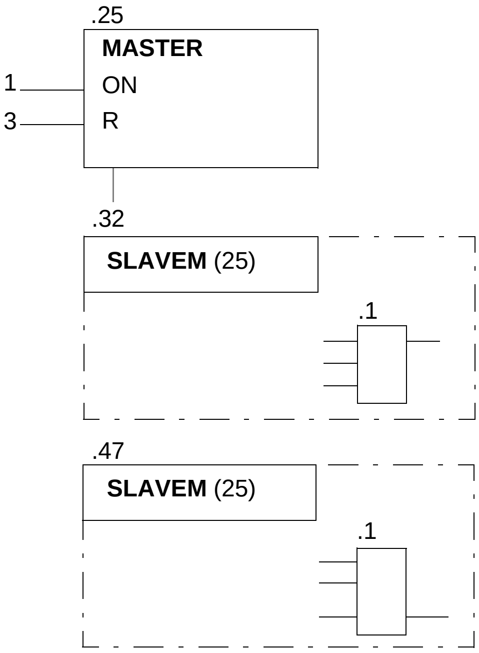


Figure 9-5. Example of the Use of Item Designations for Master Header

9.4 Blocking from the Engineering Station

A master-slave group can be blocked from an Engineering Station in the same way as a control module. The entire group is blocked or deblocked as one unit.

9.5 Modify Permission

A master-slave group is, like a control module, provided with a built-in check of the right to change the function of a master-slave group.

The output MODP (Modify Permission) from the master header indicates if Modify Permission is active or not.

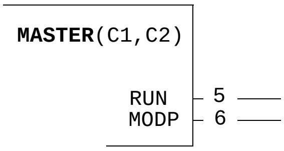


Figure 9-6. RUN and MODP Outputs from the Master Header

9.6 Master-slave in the Hierarchy

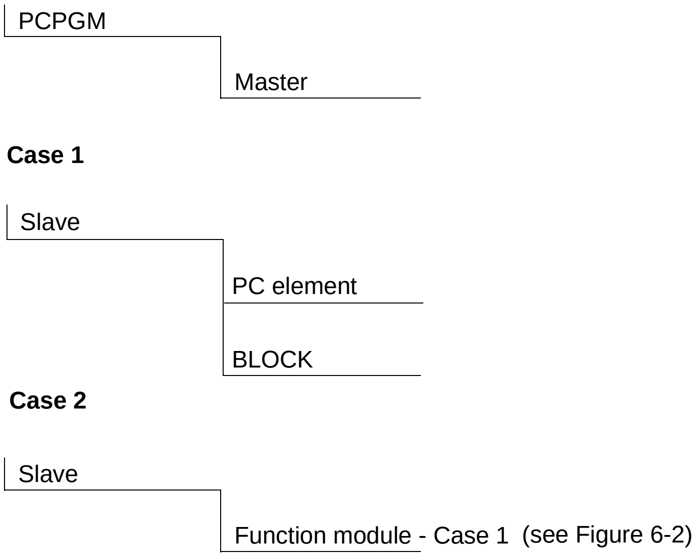


Figure 9-7. Master-Slave in the Control Hierarchy

A master must always be located directly under a PC program.

A slave may be included, with other slaves, in a function module.

PC elements and PC blocks may be included in a slave.

A slave may contain function modules.

Chapter 10 Sequence and Step Modules

10.1 General

Sequential control is the control of a process or machine performing a predetermined sequence of steps. This may be illustrated by the following example: A car is to go from its original position at A in Figure 10-2 to position B, stop, wait and then return to A.

The sequence may be described in a flow diagram according to Figure 10-1.

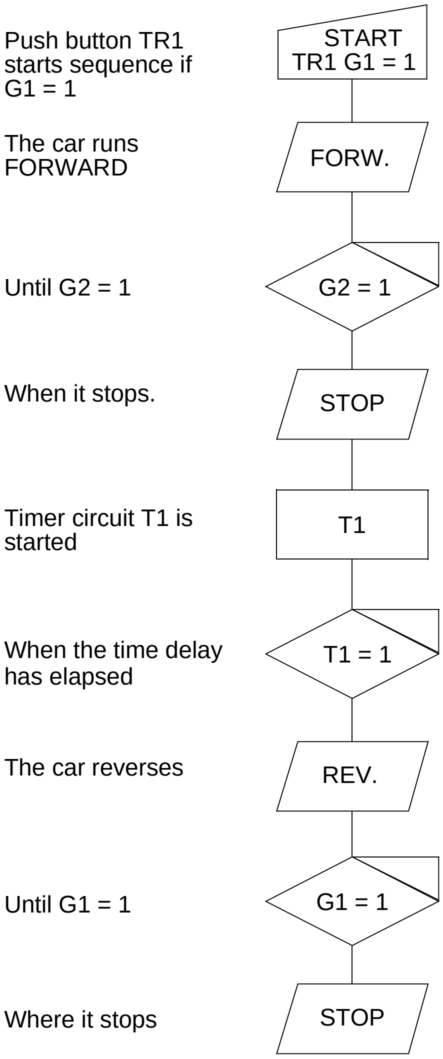


Figure 10-1. Flow Diagram for Car Sequence Control

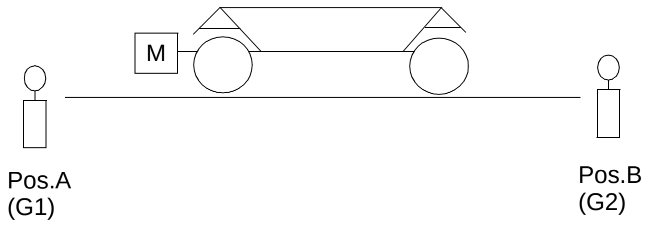


Figure 10-2. Car Control

A function diagram as shown in Figure 10-3 often gives a clearer picture of a sequence. This manner of describing the sequence is in accordance with IEC 1131-3 standard and is described in detail in Chapter 15 Applications with Sequences. Such a diagram should always be prepared as a basis for the PC diagram.

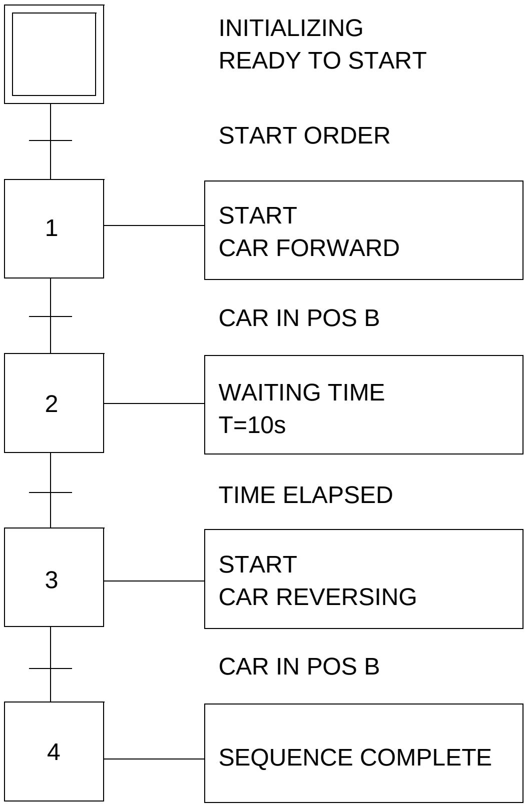


Figure 10-3. Function Diagram for Car Control

Figure 10-4 shows how the sequence is implemented in the Advant Controller 400 Series system.

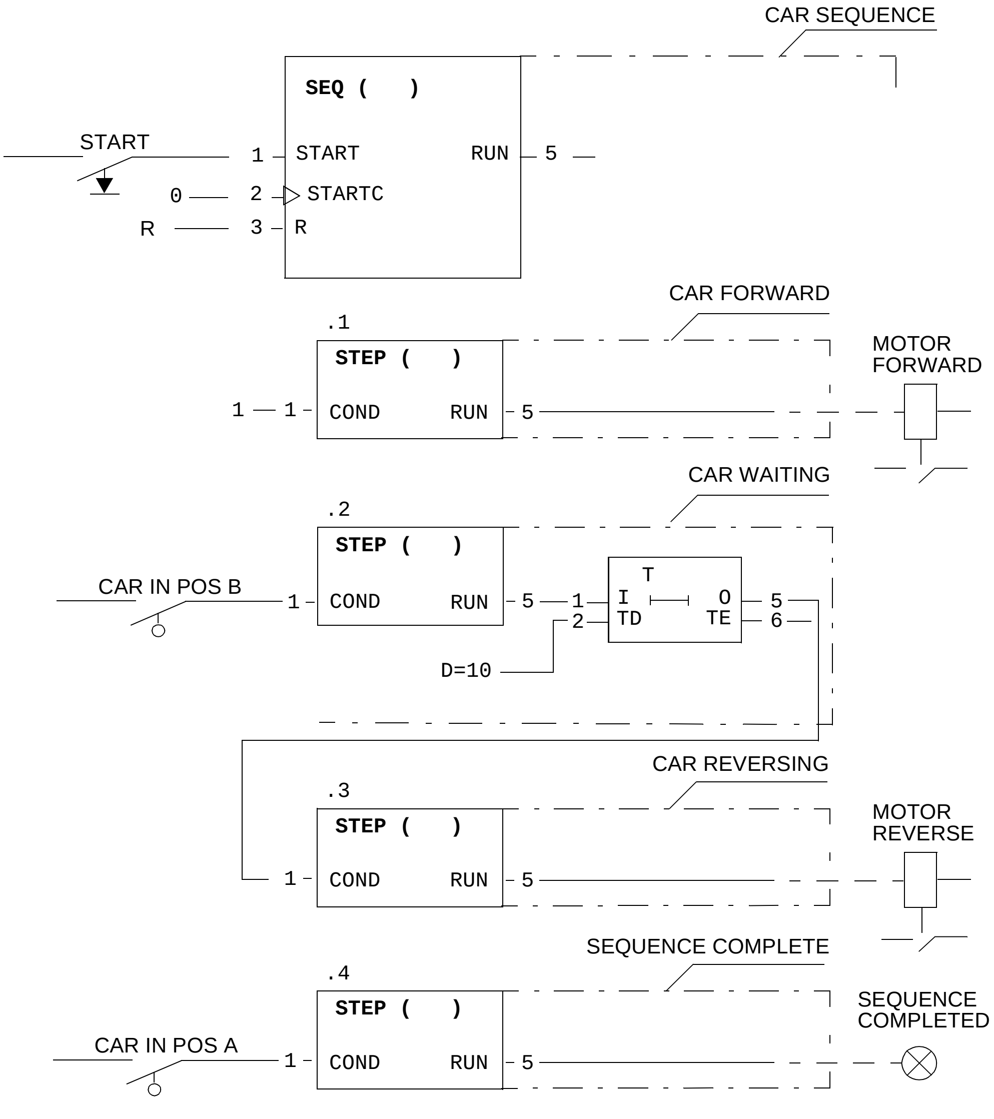


Figure 10-4. Car Sequence implemented in an Advant Controller 400 Series System

The sequence consists of a superior sequence header with the common functions for start, initializing, manual control, and so forth and a number of step modules, one for each step in the sequence. Each step module has its own step module header whose function is determined by a number of control signals.

10.2 Dividing of a Sequence into Steps

The sequence of process events is divided into a number of steps. Only one step may be active at a time. While a certain step is active, the conditions for step change are checked and a change is performed if a condition is fulfilled.

It may sometimes be advantageous to regard a process as a sequence, even if it is in itself not sequential. The result may be a simplified and well-structured control program.

10.3 Sequence in the Hierarchy

A sequence is thus composed of a number of step modules, see Figure 10-5.

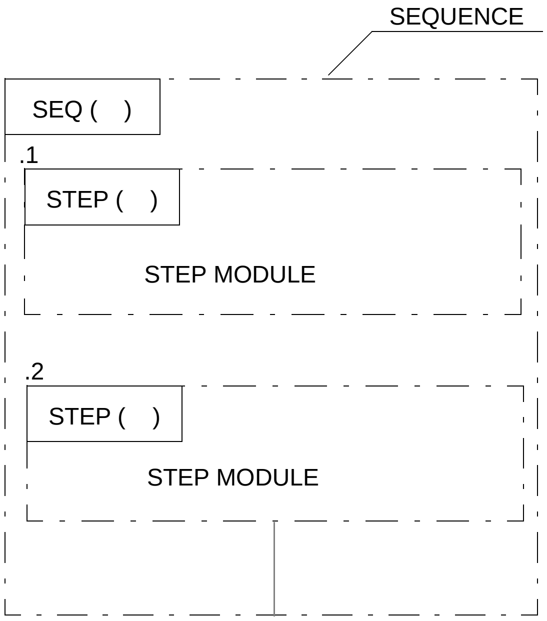


Figure 10-5. Hierarchical Subdivision

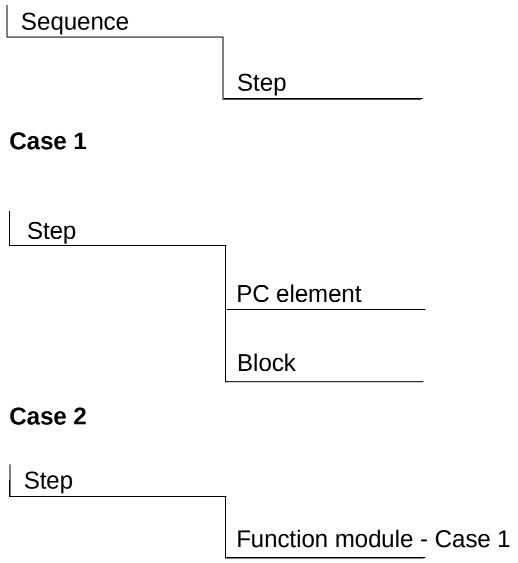


Figure 10-6. Sequence and Step in the Hierarchy

A step module may be subdivided into function modules, in which case all PC elements in the step module must be included in one of the function modules.

A sequence is an independent execution unit and may thus not be included in any other execution unit but a PC program. A sequence may be included in a function module together with, for example other sequences, control modules, or slave modules.

10.4 Function

A sequence is executed cyclically. Even if no step is active, the sequence header (SEQ) is executed once each cycle to check its control inputs. Each time the sequence header is executed and the sequence is active, the step module which at the time is active is also executed. The condition for step change is also examined. Figure 10-7 shows the principle for this.

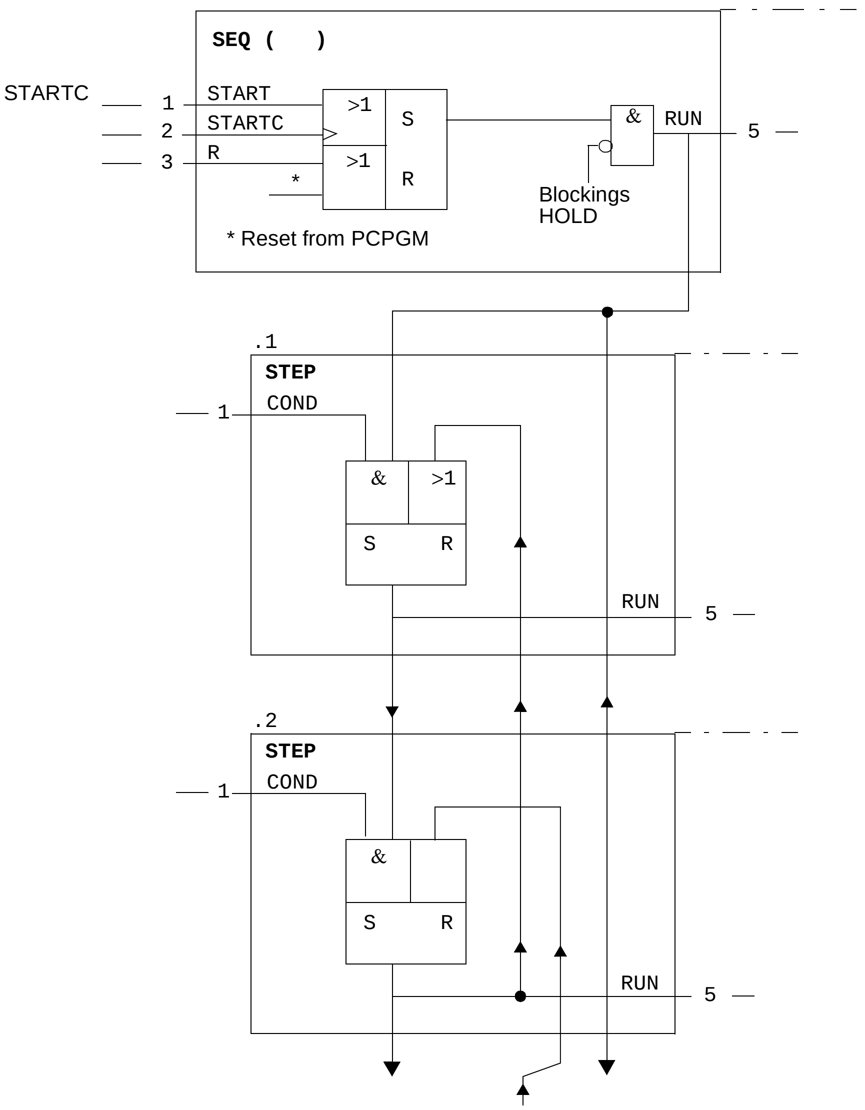


Figure 10-7. Function Diagram for Step Change

The sequence header has an internal memory function for the start signals. The memory is set using input START or STARTC and reset with the reset input R or with a superior reset signal from the PC program header.

Each step module has an internal memory function which is set when the step is activated, that is when the condition input COND is 1 and the previous step is active. This memory function is reset when the step is left or when the sequence header is reset.

Step Change

The step is normally executed in a predetermined order. First, the step header is executed, then the PC elements of the step and finally the system checks the condition for step change.

The flow diagram in Figure 10-8 shows the step change between 2 steps, step N and N + 1.

When the condition for step change from step N to N + 1 is satisfied the RUN output of step N is reset to 0 and the PC elements in the step is executed once in normal mode.

Following this, during the same cycle, the RUN output of step N + 1 is set to 1 and the PC elements of this step is executed for the first time. During this first execution of a step the condition for step change to step N + 2 is ignored. This implies that each step is activated during at least 1 cycle, that is only one step change is performed per cycle.

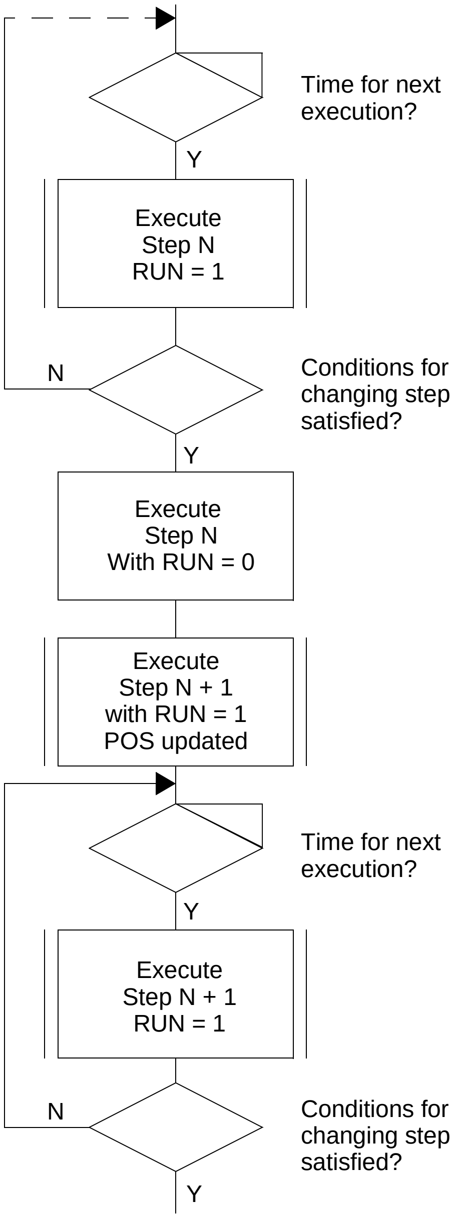

Time for next execution?
Y
Execute Step N RUN = 1
N
Conditions for changing step satisfied?
Y
Execute Step N With RUN = 0
Execute Step N + 1 with RUN = 1 POS updated
Time for next execution?
Y
Execute Step N + 1 RUN = 1
N
Conditions for changing step satisfied?
Y

Figure 10-8. Step Change

10.4.1 RUN Indications

The RUN output of the sequence header indicates that the sequence is activated and not blocked by a command from the Engineering Station. (Even HOLD affects RUN, see Section 10.5.7 HOLD below.)

The RUN output of the step header indicates that the step is active. This output may be used for direct control of an object by connecting it to the Data Base element, see Figure 10-9. This output is not affected by HOLD or by blocking commands.

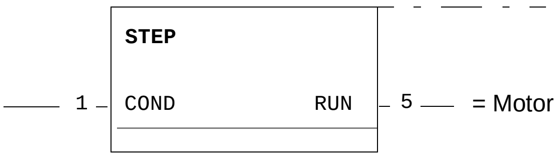


Figure 10-9. Connection of the RUN Output of a Step to the Data Base Element

In the example in Figure 10-9 the Data Base element MOTOR will be 1 as long as step 5 is active, but will reset to 0 when the step is left.

10.4.2 Step Module Input and Output

Each step module has its own input (GET) and output (PUT) elements (refer to Chapter 14 Execution). This means that

- Input from and output to the Data Base element as well as input and output for modules with differing priority (A, B or C) is performed in the normal way.
- When changing step an extra execution with the RUN output = 0 is performed activating another data transfer in the same way.
- A step which has been left is no longer executed. This includes its input and output elements. All actual data values in the step are thus retained as they were following the last execution, with those values that were obtained with RUN = 0.

Figure 10-10 shows two steps where the counter that is included in each step is connected in two different ways. In the left figure the contents of the counter will remain unchanged when the step is left, while in the right figure the counter will be cleared when the step is left since the RUN output of the step header is connected to the reset input of the counter.

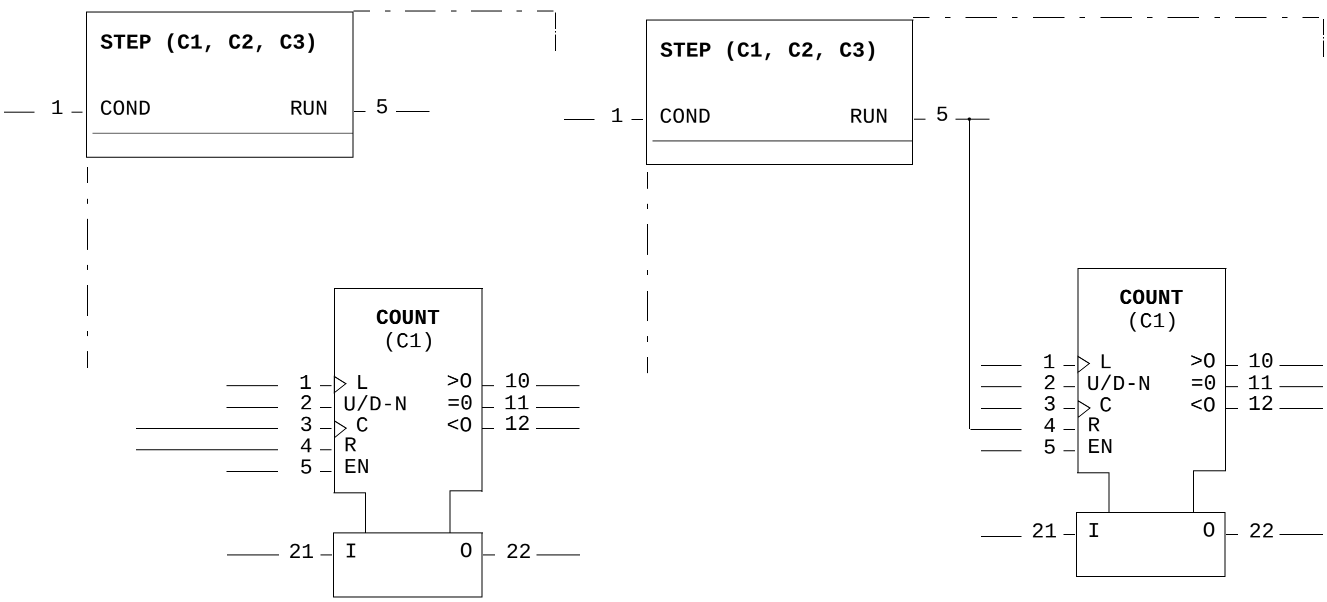


Figure 10-10. Example of Use of the Run Signal

This feature may be used when, for example, it is desired that the Data Base element MOTOR is active even after the step has been left. This can be implemented as in Figure 10-11.

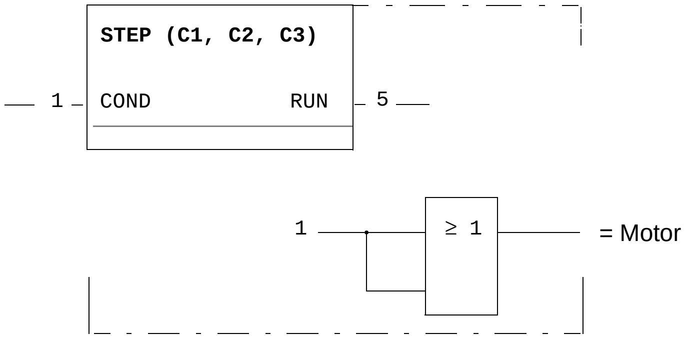


Figure 10-11. Output whose Value Remains when the Step is Left

When resetting the entire sequence, however, all output data from all steps is cleared. In our example the motor is consequently stopped when this occurs.

10.5 Sequence Header- SEQ

The sequence header, the basic symbol of which is shown in Figure 10-12, is always executed even if the sequence is inactive.

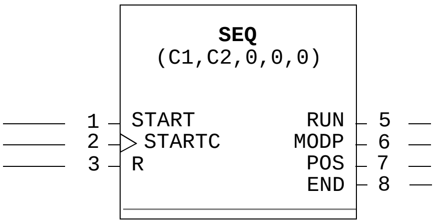


Figure 10-12. Basic Symbol for Sequence Header

10.5.1 Call

The sequence header is called with SEQ (C1, C2, C3, C4, C5). The call parameters specify:

C1	Cyclicity in milliseconds
C2	Place in the cycle time table for the cyclicity chosen
C3	Selection of manual control, Section 10.5.6 Manual Control.
C4	Selection of time supervision, Section 10.5.8 Time Monitoring.
C5	Selection of signals for MMC function, Section 10.5.9 Signals for Operator Station Functions.

C3, C4 and C5 must be specified as 1 or 0.

If C5 = 1 then C3 and C4 must also be 1.

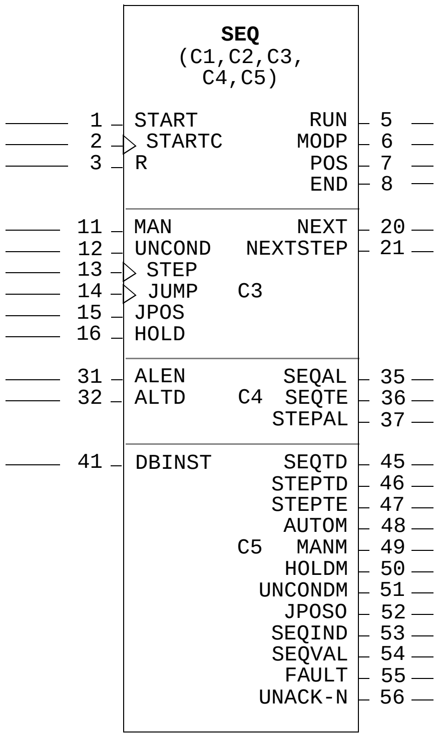


Figure 10-13. Complete Symbol for Sequence Header

10.5.2 Control

The inputs START and STARTC may be regarded as set inputs for the internal memory as shown in Figure 10-14. The STARTC input is a dynamic input, that is the memory is set to 1 when the input signal changes from 0 to 1.

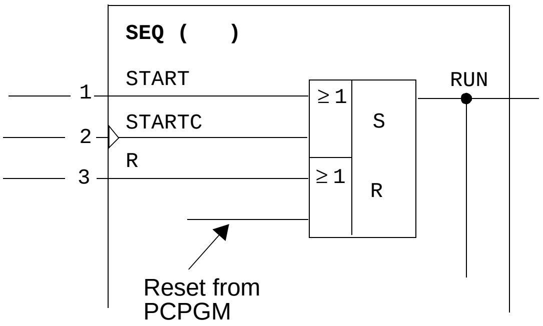


Figure 10-14. Control of the Internal Memory - START, STARTC and R

The internal memory is reset with the R input but can also be reset from the PC program header (common reset of all execution units in the PC program).

10.5.3 Blocking from the Engineering Station

A sequence may in the same way as a control module be blocked from an Engineering Station equipment.

An individual step may not be blocked since the execution unit is composed of the sequence with its steps.

10.5.4 Indications

The RUN output of the sequence header indicates if the sequence is active or not. This output is 0 if HOLD is active.

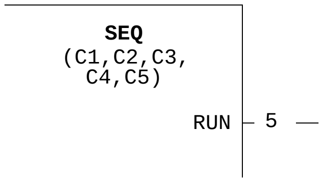


Figure 10-15. RUN Output from Sequence Header

Modify Permission

A sequence has, in the same way as a control module, a built-in check of permission to change the function of the module.

The output MODP (Modify Permission) is set to 1 if permission to change is granted.

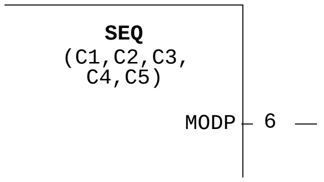


Figure 10-16. MODP Output from Sequence Header

Step Number (POS)

The POS (Position) output indicates with a number (data type I) which step is active. The value is equal to the last part of the item designation of the step.

If the sequence is inactive or if only the sequence header is active the POS output has the value 0.

When changing steps the POS output is updated during the same cycle as the step change takes place.

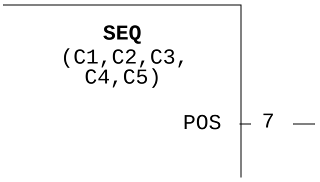


Figure 10-17. POS Output from Sequence Header

Last Step (END)

The END output is set to 1 when the last step of the sequence is activated. It is not necessary to specify which step is last when entering the program, the system detects this during execution.

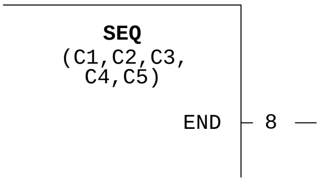


Figure 10-18. END Output from Sequence Header

10.5.5 Resetting a Sequence

A sequence is reset by activating the R input in the sequence header. If the END output is connected to the R input as shown in Figure 10-19 or if the RUN output of the last step is connected to the R input an automatic resetting is obtained.

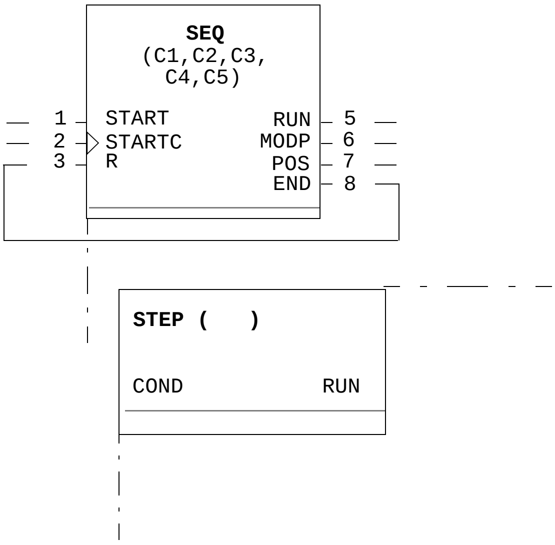


Figure 10-19. Automatic Sequence Clearing

10.5.6 Manual Control

Manual control of a sequence is selected by setting the call parameter C3 to 1. The sequence header will then appear as in Figure 10-20.

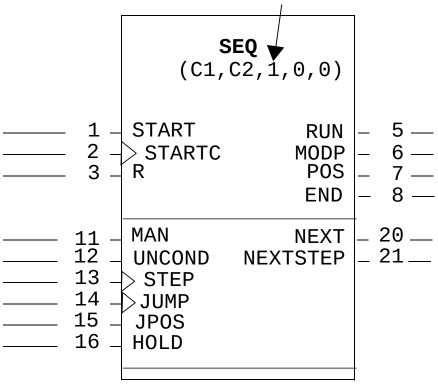


Figure 10-20. Sequence Header with Functions for Manual Control

This part of the header comprises:

- Manual stepping with or without normal step conditions.
- Unconditional jump.
- Unconditional inhibition of execution.

Figure 10-21 shows, in the form of a logic diagram, how the manual control is implemented internally.

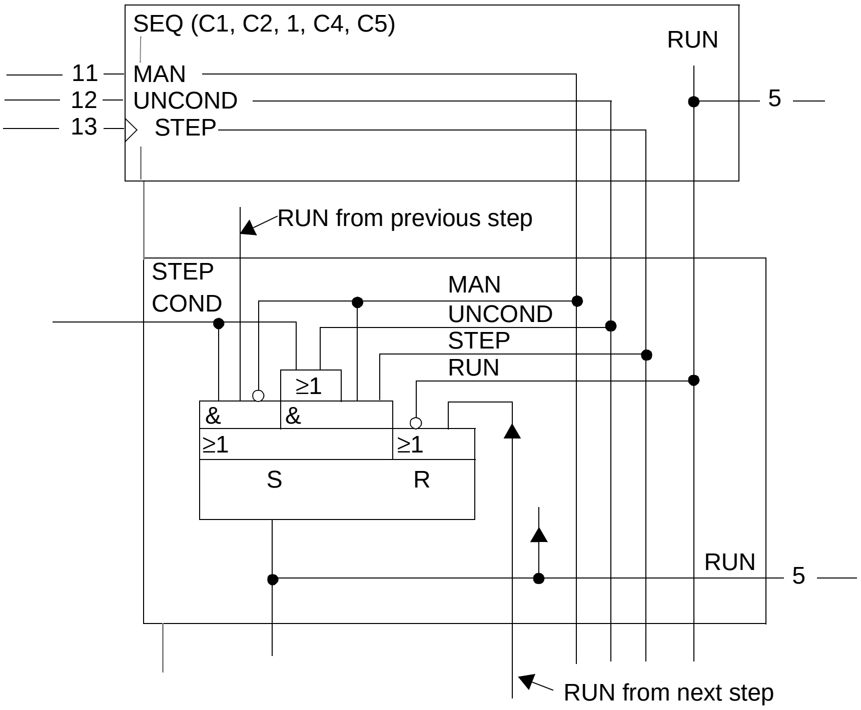


Figure 10-21. Manual Control of a Sequence

If the MAN input is 1 the automatic stepping controlled by the COND inputs of the steps is inhibited.

To permit stepping, the condition for step change must be fulfilled. This is indicated by the NEXT output of the sequence header. This output may be connected to a lamp to inform the operator that stepping is permitted. Stepping may now be initiated by activating the STEP input of the sequence header.

Using the input UNCOND (Unconditional) it is possible to override all condition inputs and in this way go through an optional number of steps.

The input JUMP initiates a jump to the step specified at the JPOS input. In the example in Figure 10-22 an unconditional jump to step 15 is obtained when the push-button is pressed.

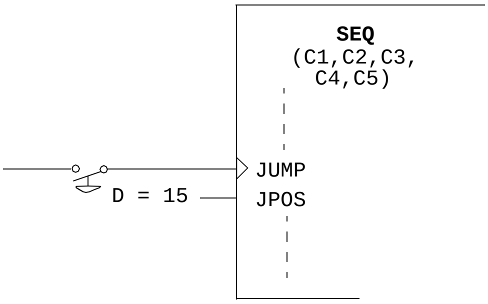


Figure 10-22. Inputs for Jump

The JUMP input is a dynamic input, that is the jump function is activated when this input changes from 0 to 1.

The jump address must be the last term of the item designation of the destination step. Address 0 denotes a jump back to the sequence header.

A jump is performed in the same manner as a step change. The JUMP input is tested when the active step header is executed. If the JUMP input is activated a step change with jump to the JPOS address is performed irrespective of the condition for normal step change.

10.5.7 HOLD

The execution of an activated step module may be interrupted temporarily by setting the HOLD input of the sequence header to 1. The execution is resumed when the HOLD input is set to 0.

Activation of HOLD has the following consequences for the sequence header:

- Its RUN output is set to 0.
- The execution of the built-in time functions are inhibited. When HOLD is deactivated the timers continue from the values they had when HOLD was activated.
- Jumps are not performed.

In other respects the sequence header is executed normally.

For the step HOLD implies that

- Execution is inhibited, that is all data calculated in the step remains unchanged.
- No new output is produced neither to the Data Base element or to other modules with different priority.
- All timers in the active step are stopped. When HOLD is released the timers are resumed from their respective values.

During HOLD the execution of the step header is inhibited and thus its RUN output is 1 during HOLD.

Objects controlled from a sequence may be controlled while HOLD is applied from another PC module whose execution may be controlled by an inverted HOLD signal.

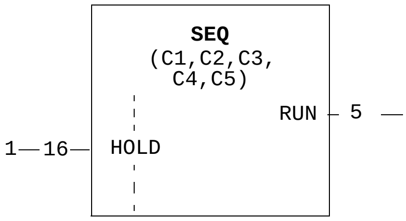


Figure 10-23. HOLD Input

10.5.8 Time Monitoring

A delay PC element is introduced into the sequence header if the call parameter C4 = 1, see Figure 10-24. The delay PC element is of the integrating type like the PC element TONRET. This function may be used for time supervision of the sequence.

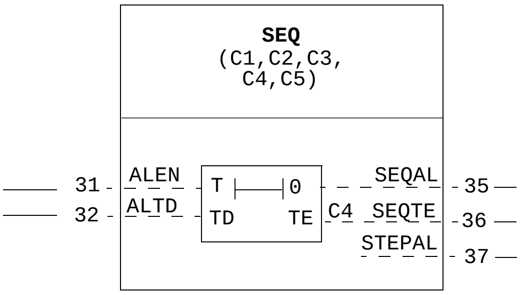


Figure 10-24. Sequence Header with Time Supervision Selected

Time measurement is performed as long as the input ALEN (Alarm Enable) is 1. The delay time is determined with the input ALTD (Alarm Time Delay) to a maximum time of 24 hours. The accumulated time is available at the output SEQTE (Sequence Time Elapsed). The output SEQAL (Sequence Alarm) indicates when the delay time has expired.

Timing stops when HOLD is activated.

The elapsed time is cleared only when the sequence is reset.

The sequence header also contains a STEPAL output which is set to 1 if the time delay of any of the active steps has expired. This function is described in more detail in Section 10.6.4 Time Function.

10.5.9 Signals for Operator Station Functions

TheOperator Station Function is selected with the parameter C5. If this part is selected the previous two parts also have to be selected.

The signals in this part reflects the operational state of the sequence and is used to present the state on a screen of the Operator Station.

Refer to the PC element data sheet for a more detailed description.

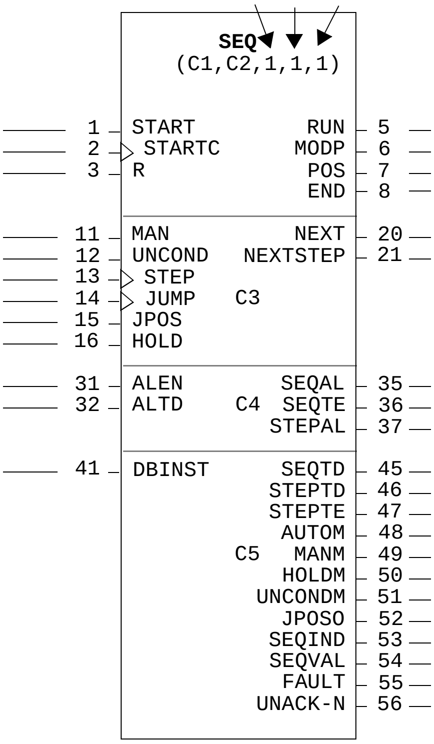


Figure 10-25. Parts of Sequence Header Selected with Call Parameters

10.6 Step Module - STEP

The basic symbol of the step module with its condition input (COND) and status output RUN is shown in Figure 10-26.

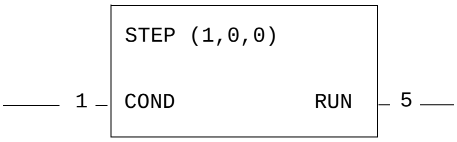


Figure 10-26. Basic Symbol of Step

10.6.1 Call

The step header is called with STEP (C1, C2, C3). The call parameters have the following significance:

C1 Selection of time supervision, with value 1 or 0

C2 Jump control, 0 - 4 jump destinations

C3 Selective resetting of steps, 0 - 4 steps

The complete symbol of the step header with auxiliary functions is showed in Figure 10-27.

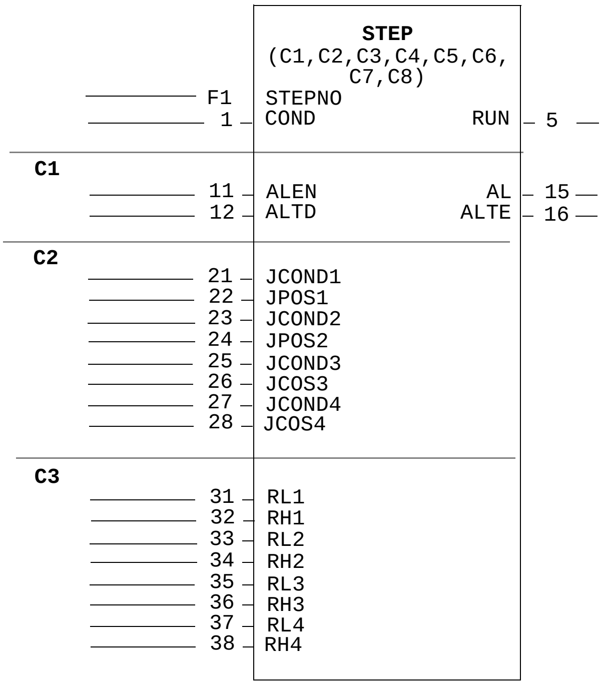


Figure 10-27. Complete Symbol for Step Header

10.6.2 Condition Input (COND)

The condition for change of step may be a boolean variable, which then must be connected to the COND input as in Figure 10-26.

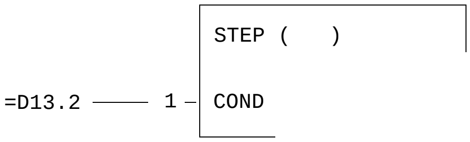


Figure 10-28. COND Input

Composite conditions may be formed using PC elements in the usual way. The PC elements forming the condition then belong to the previous step module.

10.6.3 RUN Indication

The RUN output of the step header indicates if the step is active. The output may be connected to other PC elements or to the Data Base element.

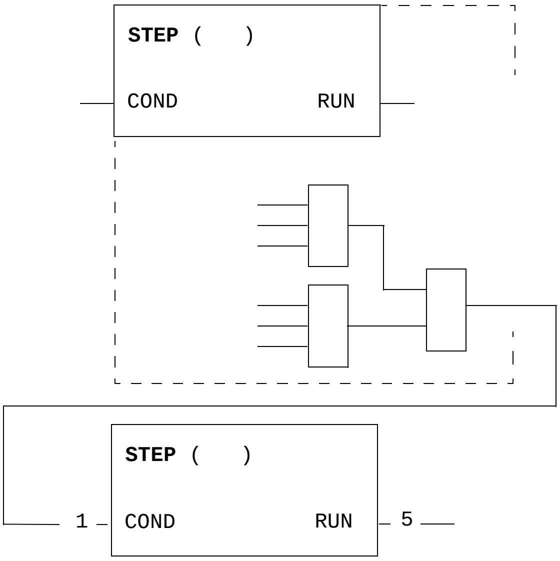


Figure 10-29. Example of Connection of COND Input

10.6.4 Time Function

The time function of the step header is selected with the call parameter C1. This time function is analogous to that of the sequence header.

The delay time is specified with the input ALTD and the maximum time is 24 hours.

If time supervision and Operator Station part has been chosen for the sequence header the values of the signals AL, ALTE and ALTD are transferred to the outputs STEPAL, STEPTD and STEPTD respectively for the active step.

If the input ALEN of a step header is connected to a constant 1 the timer circuit is started when the step concerned is started. This may be used when a step is to be active a specified time as shown in the following example of a sequence.

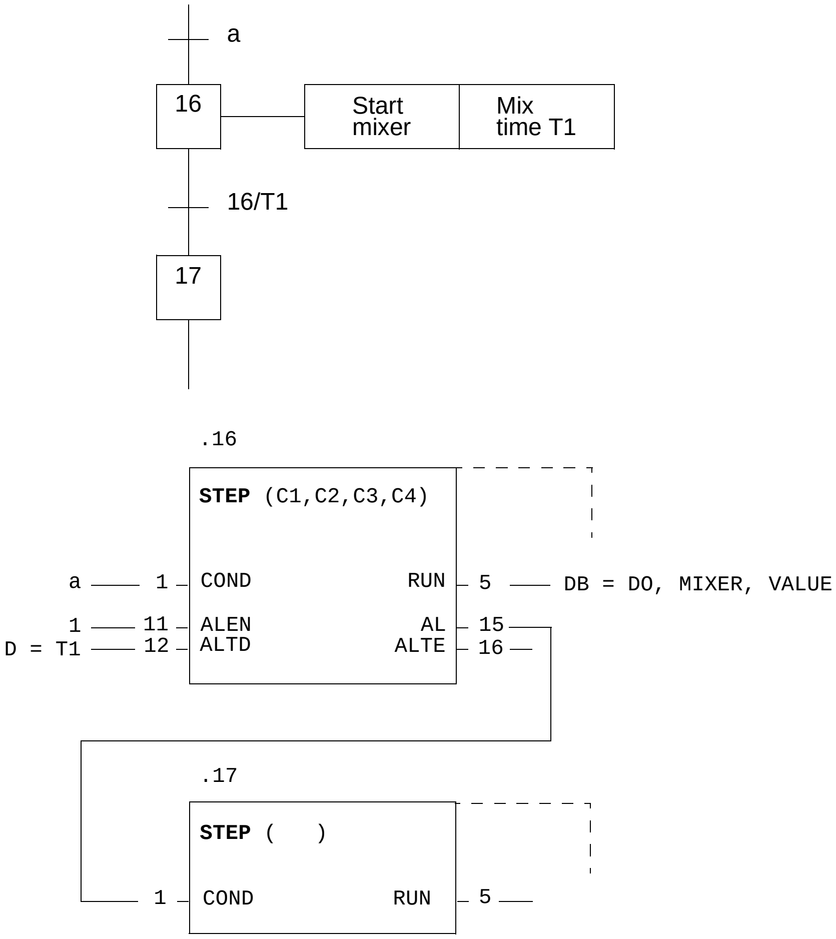


Figure 10-30. Example of Use of the Built-in Time Function

In step 16 a mixer is to start and run time T1.

The timer circuit is stopped if HOLD is activated and is cleared when the step is left.

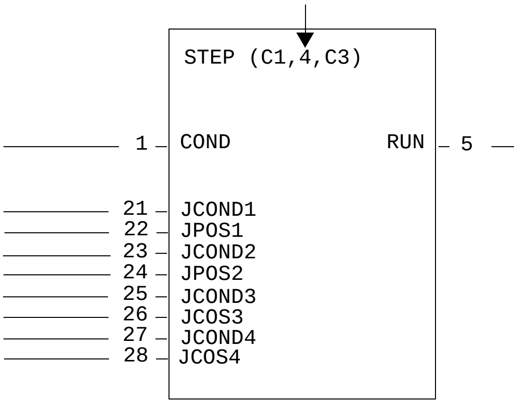


Figure 10-31. STEP with 4 Conditional Jumps

10.6.5 Conditional Jump

The parameter C2 specifies for each step how many additional branches possible to select after this step. A maximum of four jump targets may be specified in addition to the normal consecutive step.

The example in Figure 10-32 shows a sequence where it is desired to jump from step 5 to step 14 if the condition a is satisfied. If not, a step change to step 6 is performed if condition b is satisfied.

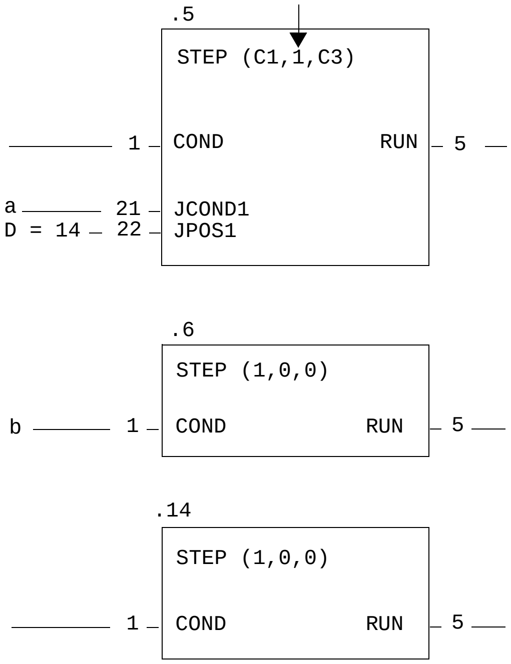


Figure 10-32. Example of Jumps

For step 5 C2 is specified to be 1 and the condition a is connected to the input JCOND 1. The input JPOS1 is connected to a value containing the jump address as a constant or operational parameter.

During execution the JCOND inputs are examined before the condition for step change.

A conditional jump may be forward or backward in the sequence. If the jump address is 0 the sequence jumps backward to the sequence header. If a non-existent address is specified the sequence remains in the step from which the jump was to originate.

10.6.6 Selective Reset

The normal resetting of a sequence is done via the Reset input of the sequence header. It may sometimes be necessary to perform a selective reset of a number of steps in the sequence.

Using the call parameter C3 the number of groups of steps to be selectively reset from a particular step may be specified. The reset is performed when the step is left.

Figure 10-33 shows the symbol of the step header with C3 = 4, that is equal to the maximum number of groups. For each group the start and stop address is specified. All steps from the start address to the stop address inclusively according to the documentation order will be reset.

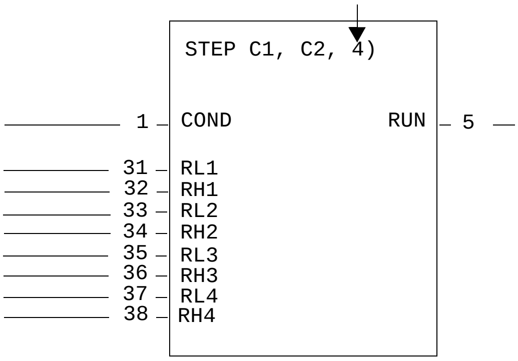


Figure 10-33. Step Module Header with C3 = 4

RL (Reset Low) specifies the start address and RH (Reset High) specifies the stop address. If only one step is to be reset, the address of that step is specified as both high and low reset address.

In the following example it is desired that step 8 and step 12 through 26 be reset when step 32 is left. Figure 10-34 shows how this is specified.

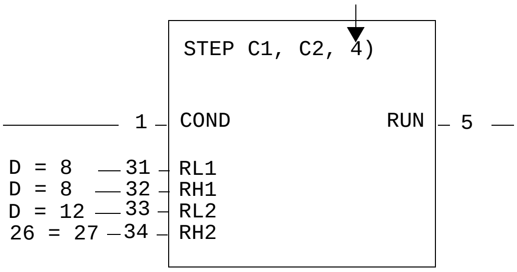


Figure 10-34. Example of Selective Resetting

- If a step with address RH does not exist the resetting is finished when the last step of the sequence has been reset.
- If address 0 is specified as RL all steps in the sequence are reset. The execution then continues with the following step or the step designated as the jump target.
- The documentation order decides which steps are to be reset, not the running enumeration.

Chapter 11 PC Block - BLOCK

11.1 General

The PC block is a tool to switch execution of a part of an execution unit (master, step or slave module) on or off.

PC blocks are used if there are functions within an execution unit that need only be performed on particular occasions.

The processor load of the system can also be reduced by controlling a group of PC elements so that they are only executed when necessary. This control is performed with the input ON of the block header, see Figure 11-1.

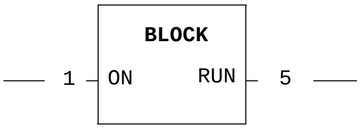


Figure 11-1. Block Module BLOCK

Figure 11-2 shows an example of the use of PC block. The block is identified (item designation) in the same way the other PC elements in the PC module are.

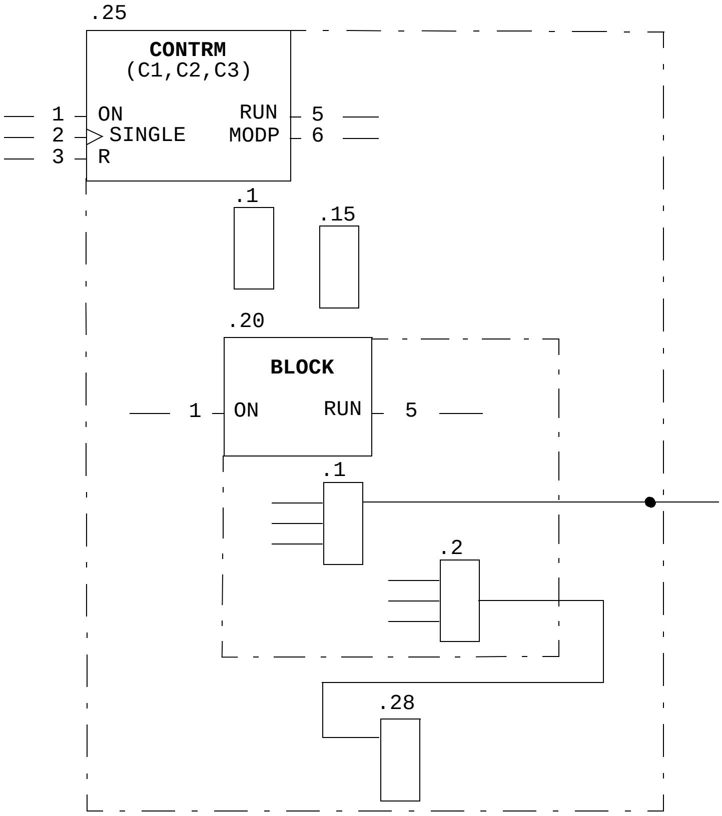


Figure 11-2. Example of the Use of PC Block

Blocks may be included in function modules, control modules, slave modules and steps together with PC elements. A block may only contain PC elements.

When a block is switched off (ON = 0) the following occurs:

- Data calculated in the block remain unchanged.
- Output of data to the Data Base element and to modules with different priority continues as long as the execution unit to which the block belongs is executed.

When an execution unit is reset the block modules in the execution unit are affected whether they are on or off.

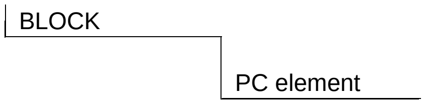


Figure 11-3. BLOCK in the Hierarchy

Chapter 12 The PC Element Types

12.1 General

The PC elements in Advant Controller 400 Series may be divided into element types where each type performs one type of function. The PC element types are:

- Structure Elements
- Combinational Elements
- Memory Elements
- Time Elements
- Register Elements
- Switch Elements
- Multiplexer Elements
- Code Converter Elements
- Arithmetic Elements
- Counter Elements
- Comparator Elements
- Function Generators
- Feedback and Filter Elements
- Panel Elements
- Data Set Elements
- MasterFieldbus Communication Elements
- Hardware Module Handling Elements
- Supervision Elements
- Application Adapted Elements
- Text Handling Elements
- Functional Unit Elements
- Table Handling Elements

The PC element libraries for the different Advant Controller 400 Series products contain different subsets of all the PC elements.

Each PC element is described in *PC Elements Advant Controller 400 Series Reference Manual.*

Chapter 13 PC Programming

The first part of this manual describes the basics of the PC programming language. This second part describes how to design a PC program by forming a structure and combining the PC elements.

Some basics that apply to a PC program are:

- A system for identification of PC elements and PC modules
- Rules for connection of the PC elements
- Design rules for the PC diagram

13.1 Identification (Item Designation) of PC Elements

As previously described the identification system for PC programs, PC modules and PC elements in a Advant Controller 400 Series system is built up in an hierarchic form, similar to the division of a book into chapters, sections and paragraphs. Figure 13-1 shows an example of the principle of identification. This principle may be more easily understood if the program module division is drawn in structure tree form as shown in Figure 13-2.

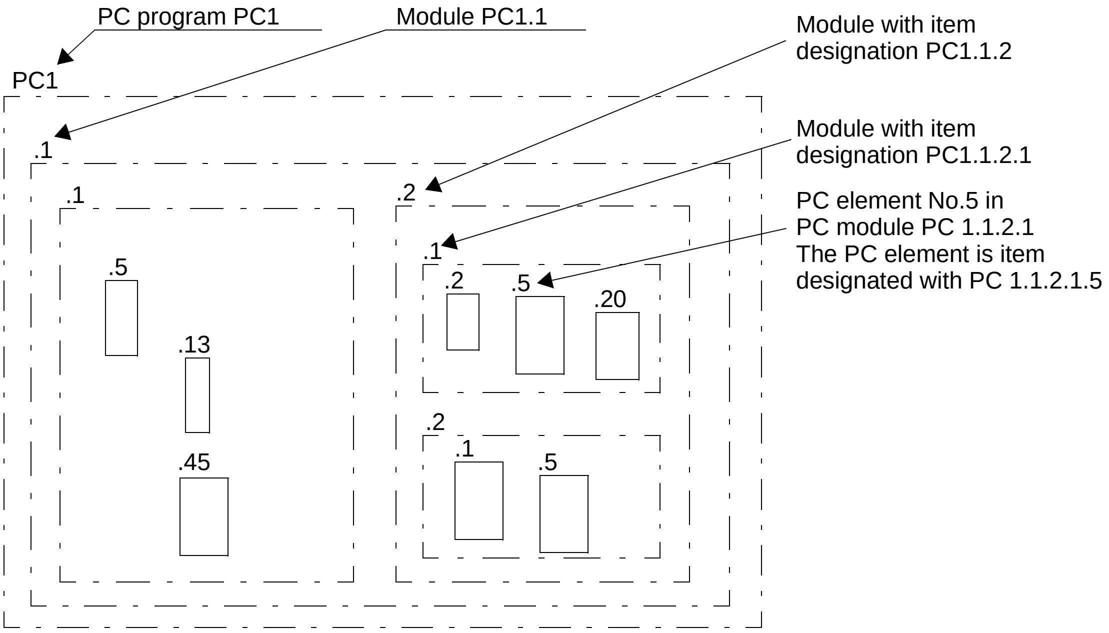

PC program PC1
Module PC1.1
Module with item designation PC1.1.2
Module with item designation PC1.1.2.1
PC element No.5 in PC module PC 1.1.2.1 The PC element is item designated with PC 1.1.2.1.5
PC1
.1
.1
.5
.13
.45
.2
.1
.2
.5
.20
.2
.1
.5

Figure 13-1. Example of Division into Modules in a PC Diagram with Item Designations

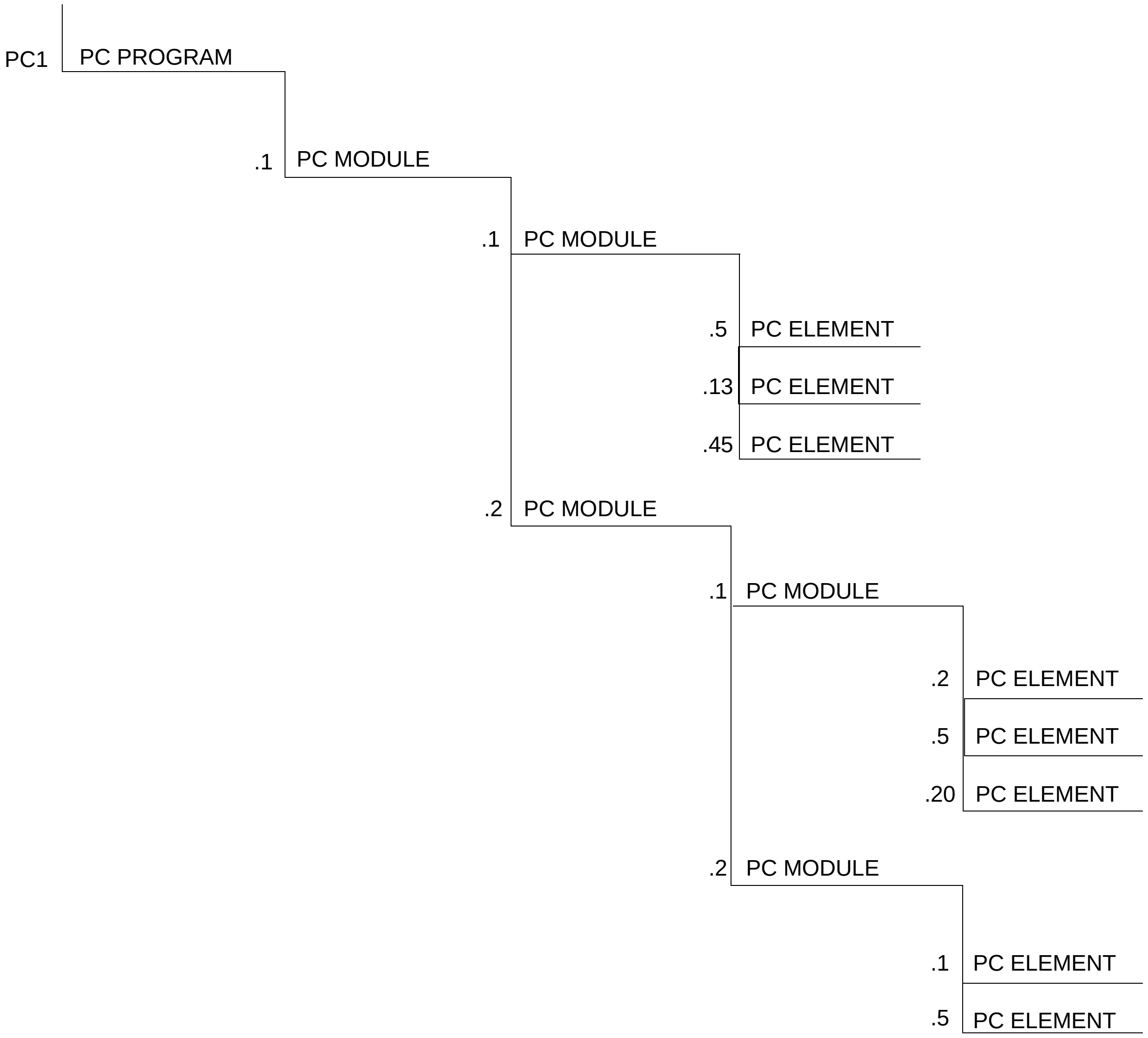


Figure 13-2. Division into Modules in Structure Tree Form.

PC Program

At the highest level, the PC program level, the PC programs are designated PC1, PC2, PC3 and so on. The division at this level is normally governed by the division of the entire plant.

PC Modules

Each PC program is divided into one or more PC modules which have item designations PC1.1, PC1.2 and so on in PC program PC1, and PC2.1, PC2.2 and so on in PC program PC2 and so forth. On the next lower level the designations are PC1.1.1, PC1.1.2 and so forth. The maximum number of levels in a PC program including the program and PC element levels is 9. Each level except the PC element level (see below) may be further divided into 1 to 999 new PC modules.

PC Elements

The PC elements in a PC module are given item designations ranging from 1 to 255, that is a PC module can contain a maximum of 255 PC elements.

The common part of the item designation for a PC module or a PC element is printed above the upper left hand corner preceded by a dot (.) except for the highest level (PC programs).

In the example in Figure 13-3 the PC element gets the complete item designation PC1.2.25.

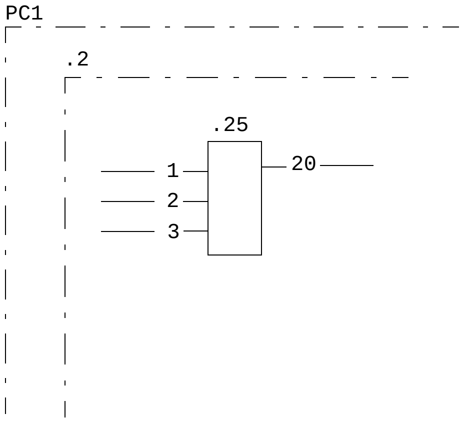


Figure 13-3. Item Designations

The common part of the item designation for a diagram page is printed uppermost on the page, see Figure 13-4.

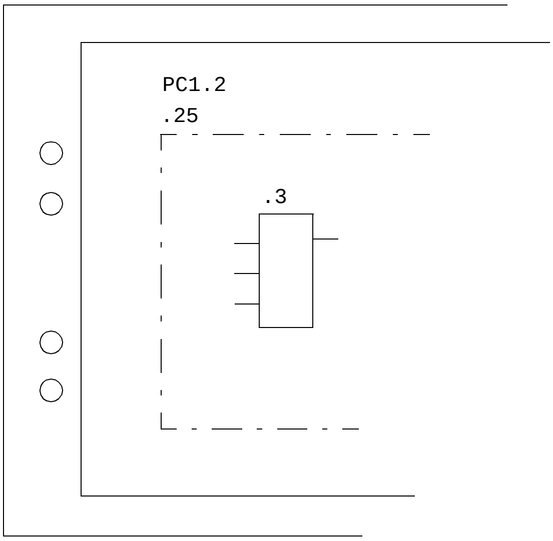


Figure 13-4. Item Designations

This item designation of PC elements and PC modules is used on program entry and editing, but is also used as a reference when debugging and testing the program.

13.2 Identification of PC Element Terminals

A terminal of a PC element is identified by the item designation of the PC element followed by a colon (:) and the terminal number. If a symbolic name exists for the terminal this may be used instead of the terminal number. The output of the PC element in Figure 13-3 is completely identified by the item designation PC1.2.25:20 and is identified within the PC module by 25:20.

13.3 Connection of PC Elements

In the PC diagram a connection between two PC elements is indicated with a line between the corresponding PC element terminals in the same manner as in a conventional logic diagram.

This connecting line denotes a data exchange between the two PC elements and is always specified in this way irrespective of the data type. If the PC elements are located on different diagram pages the PC item designation of the PC element with the output is printed at the connecting line of the PC element with the input. A cross reference is also printed for the diagram pages.

Connections to the Data Base element, to constants and to operational parameters are specified with a reference at the connecting line for the output or input concerned.

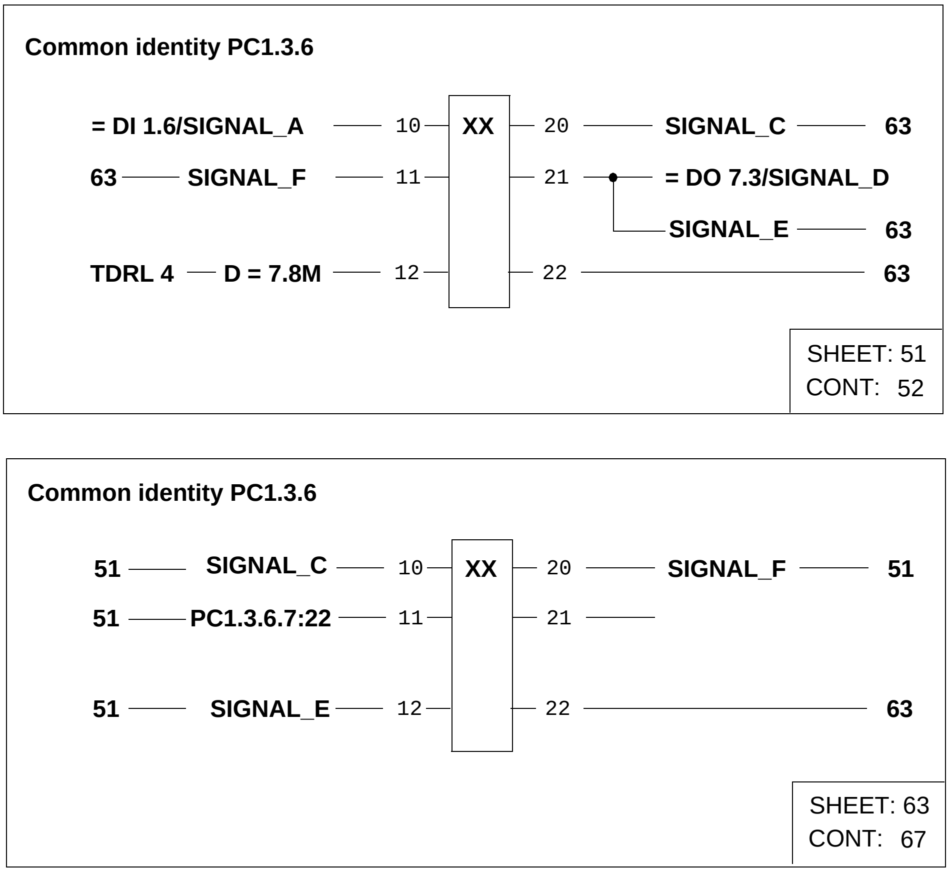


Figure 13-5. Example of Connections

When entering a PC program from a draft PC diagram the output variable, Data Base element, constant or operational parameter that is to be connected to each input is given. In other words the source of each input variable is given, see Figure 13-6.

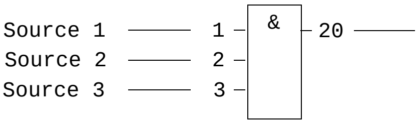


Figure 13-6. AND-element AND(3)

The consumer(s) of an output variable is given at program entry only if the output variable concerned is connected to one or more Data Base elements.

A symbolic name for an output variable may be defined when the PC element concerned is connected.

An output variable may be connected to several PC elements within and between different PC modules without restrictions. Refer to Section 13.3.4 Interconnection of PC Programs concerning connection between different PC programs.

When connecting the receiving PC elements one refers either to the item designation of the output variable (designated .4:20 in the example in Figure 13-7) or to its symbolic name (START in the example).

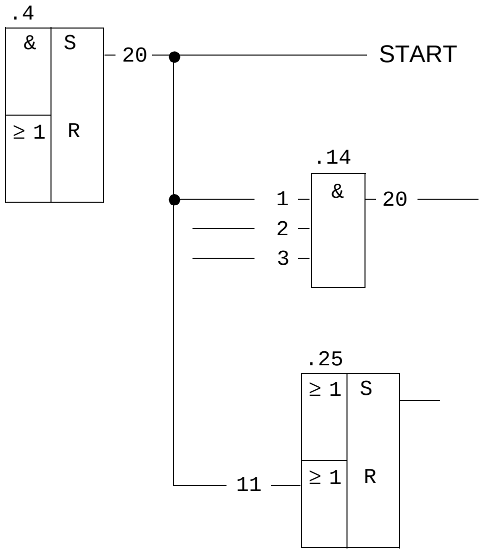


Figure 13-7. Example of Item Designations and Symbolic Name

NOTE

This name must have been declared before it is used as a reference.

Feedback

An output variable of an PC element may be returned to an input of the same PC element or to an input of an earlier PC element in the same or in another module.

A signal thus fed back is always delayed one execution cycle. In the example in Figure 13-8 the counter is not reset until the execution cycle following the one when the output > = HL becomes 1.

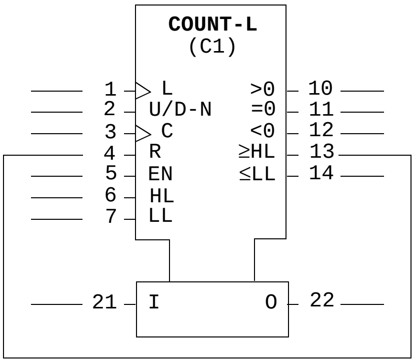


Figure 13-8. Feedback within the Same PC Element

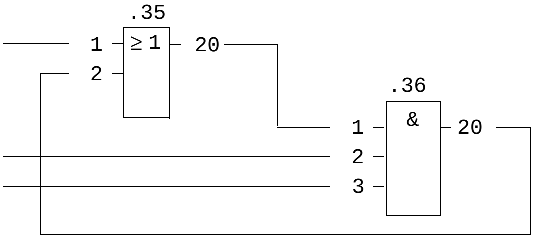


Figure 13-9. Feedback across Several PC Elements

Inverted Signal

A logical (Boolean) variable may be inverted when connected to an input terminal. This is indicated in the PC diagram by a small circle at the corresponding terminal. When connecting a terminal, inversion is obtained by a preceeding minus sign.

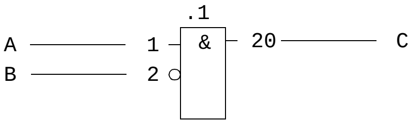


Figure 13-10. Inverted Signal

Inversion to the Data Base element

If a variable connected to a Data Base element is to be inverted, an inversion element must be used.

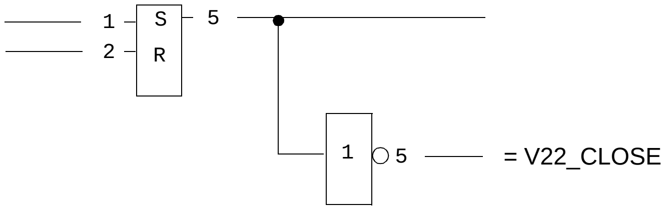


Figure 13-11. Inverted Signal to a Data Base Element

Dynamic Input

A dynamic input is a functionality of the PC element and hence is connected as any other input.

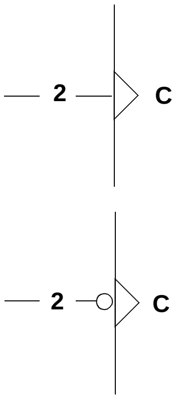


Figure 13-12. Dynamic Input

Data Type Checking

Data exchange or connection between two PC element terminals is permissible only if both terminals have the same data type. On entry and change the Engineering Station checks the respective data type and generates an error message if they are not the same.

If it is necessary to connect inputs and outputs with differing data types, for example in arithmetic expressions, conversion elements must be used.

In Figure 13-13 an output from an ADD element with data type I must be connected to a multiplication element with data type R. The conversion element CONV must then be used to convert from type I to type R.

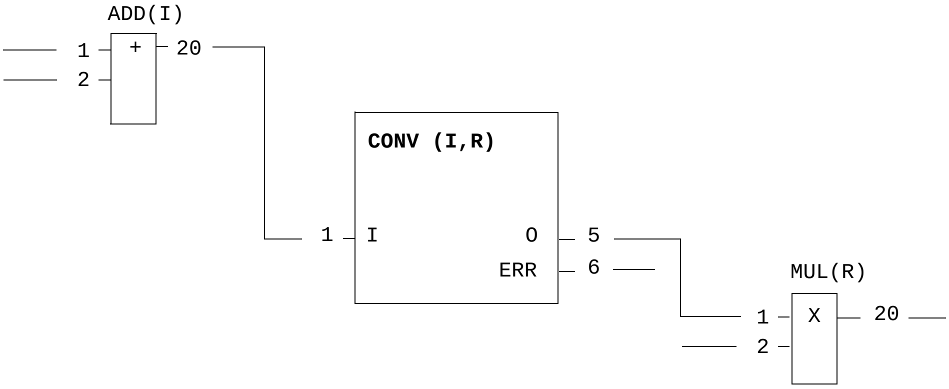


Figure 13-13. Conversion of Data Type

Interconnection of Outputs

Outputs from two or more PC elements may not be connected to a single output variable. The Engineering Station checks for this condition on entry and editing and generates an error message if an attempt is made to connect two or more output terminals.

It is, however, possible to connect to the same data item in the Data Base element. This is meaningful only when two or more sections of a program are executed alternately but affects the same outputs in the plant.

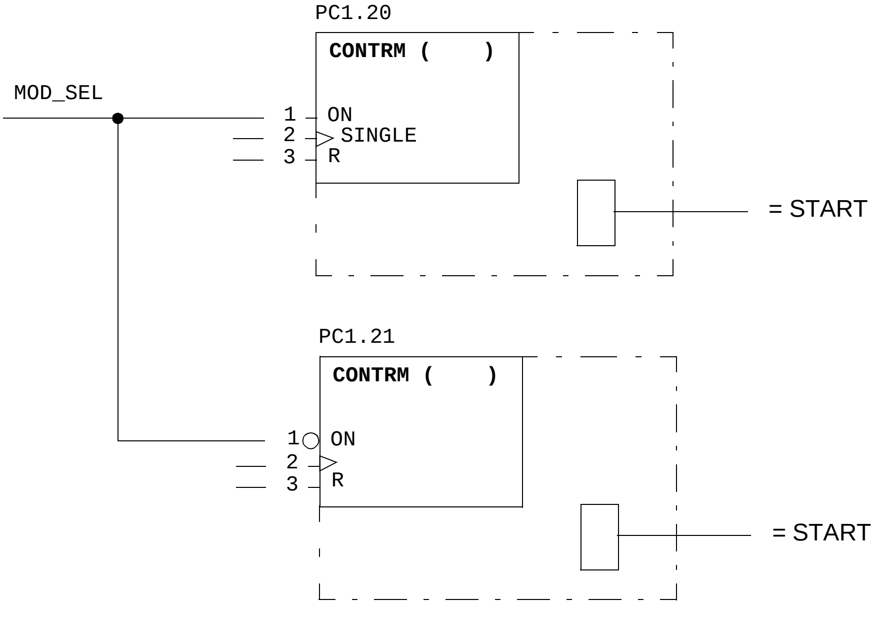


Figure 13-14. Connection of Several Outputs to the Same Data Base Element

Connection to Group Data

Figure 13-15 shows an example of connection to group data. A function generator FUNG- 1V obtains its XTAB and YTAB values from two register elements REG-G for group data.

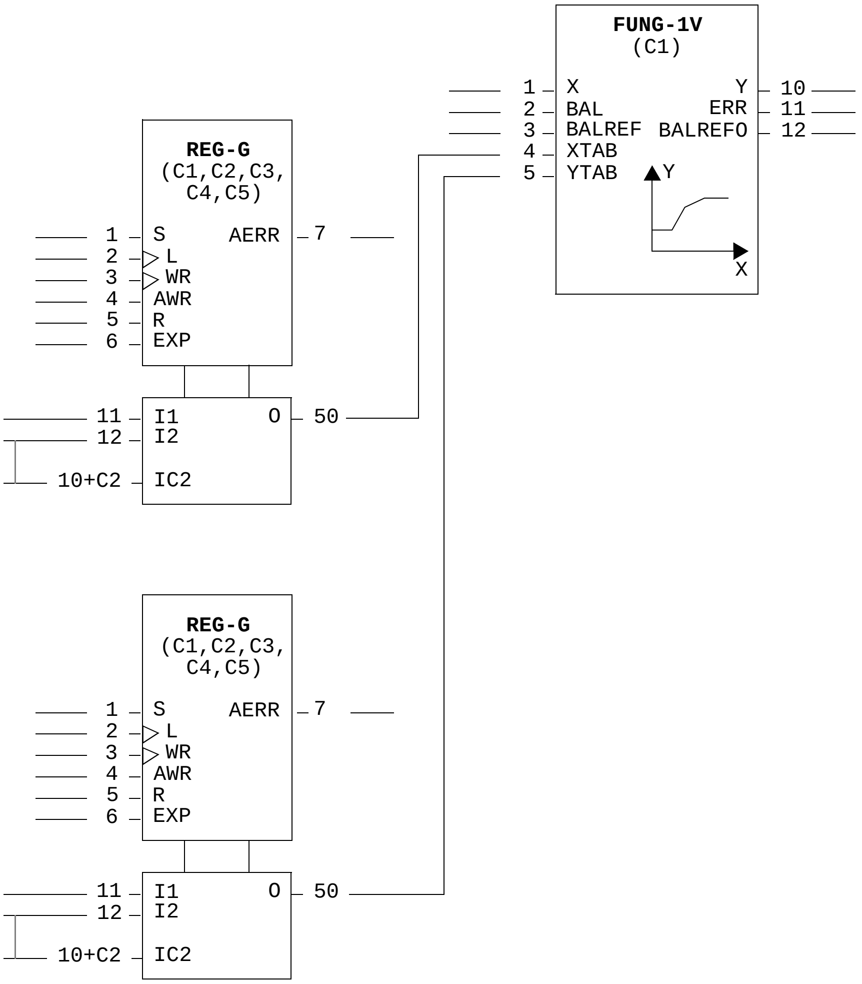


Figure 13-15. Connection of Group Data

When connecting group data both data type and the number of data in the group must match between output and input. The data output of the register element therefore has the form < GND > where

G denotes group data

N denotes the number of data items in the group

D denotes the data type

A register element for 10 real numbers will thus get the data type denotation G1OR for its data output in the PC list.

13.3.1 Connection to Data Base Element

A connection to a Data Base element is made by referencing the item designation or symbolic name. This applies to both input and output variables.

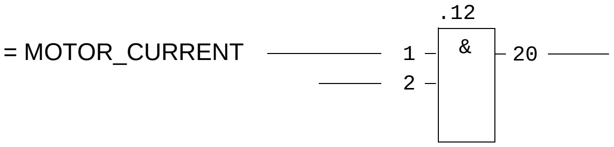


Figure 13-16. Input Connected to a Data Base Element

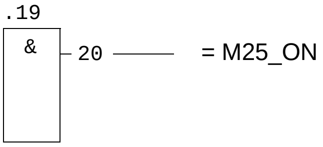


Figure 13-17. Output Connected to a Data Base Element

When several PC element inputs use the same Data Base element as input source reference may be made either to the Data Base element itself or to the first user of that element.

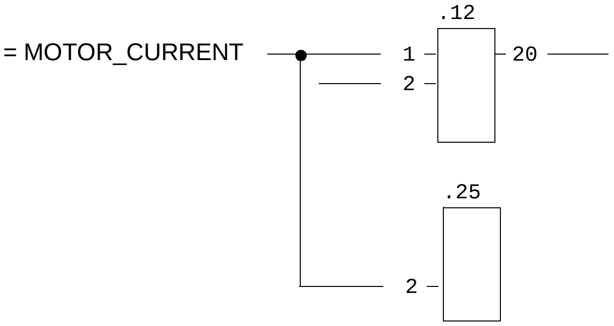


Figure 13-18. Several Inputs Connected to a Data Base element

When an output variable is to be connected to several Data Base elements each connection is automatically assigned its own output element (see Chapter 14 Execution). Fan-out thus takes place at the PC element output concerned, see Figure 13-19.

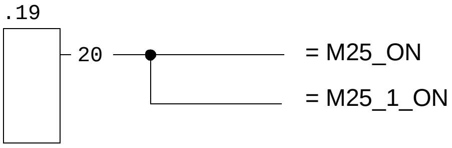


Figure 13-19. Output Connected to Several Data Base Elements

At program entry time the connections are specified in the ordinary way.

An output variable connected to a Data Base element may also be connected as an input variable to another PC element input. When entering the connections shown in Figure 13-20 the item designation of the output variable or its symbolic name is given.

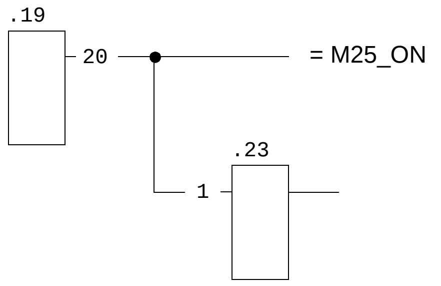


Figure 13-20. One Output Variable connected both to a Data Base Element and to a PC Element Input

If, instead, reference is made to the Data Base element when connecting the input the diagram will appear as in Figure 13-21. In this case the new value of the output signal will be available one execution cycle later.

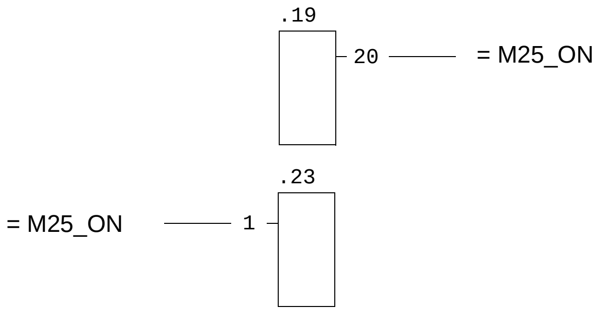


Figure 13-21. Connection of an Output Variable to an Input via a Data Base Element

13.3.2 Connection to Operational Parameters

The value of an operational parameter is specified by connecting the corresponding PC element terminal to a fixed value corresponding to the parameter value. In the example in Figure 13-22 the input is connected to a data item with the value D = 10 (s).

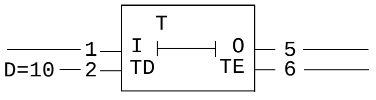


Figure 13-22. Connection to Operational Parameter

Connection to a multi-use operational parameter (refer to Section 4.7.3 Operational Parameters) is made by referring to a PC element input that uses the operational parameter or by using its symbolic name.

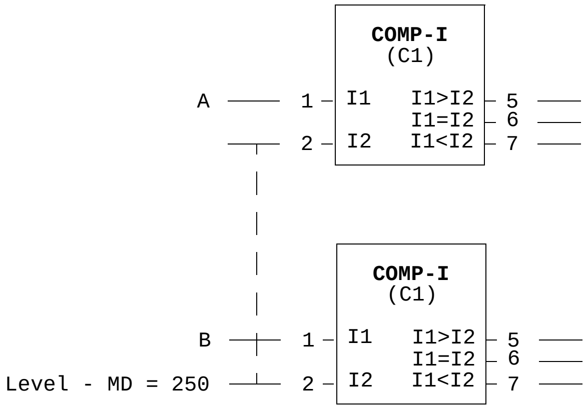


Figure 13-23. Connection to Multi-use Operational Parameter

13.3.3 Connection to Constants

Connection to a constant (a data item with a constant value that cannot be changed) is made in the same way as for an operational parameter.

Figure 13-24 shows a PC element for calculation of the square root of a number I. The constant K is the scale factor and has the value CD = 2.5 in the figure. The notation CD indicates that the data item is to be stored in such a way that it cannot be changed with the programming command Modify Value.

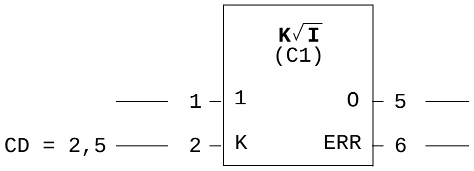


Figure 13-24. Connection to a Constant

A constant may also be common to several terminals. This is indicated in the documentation by MCD = ...

A Boolean constant may have the values 1 or 0, see Section 4.7.4 Constants. When connecting to these constants, CD = need not be given but it suffices to write 1 or 0.

An example: A counter is to be used in a particular application to count downwards and the enable input is always to be enabled. In the PC diagram 0 and 1 respectively is written in front of these inputs.

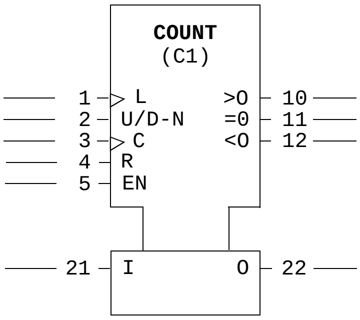


Figure 13-25. Connection to Boolean Constants

If it is necessary to have the option to, for example, turn off the execution of an execution unit the control inputs must be connected to operational parameters, that is D = is specified in the PC diagram and at entry according to Figure 13-26.

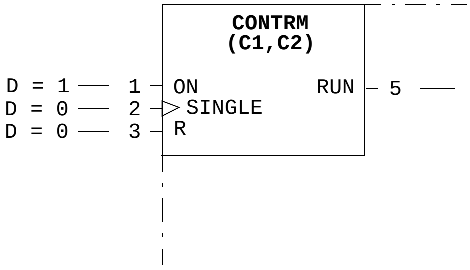


Figure 13-26. Connection of Control Inputs of a Module Header to Operational Parameters

13.3.4 Interconnection of PC Programs

Data exchange between PC elements in different PC programs must be performed via a Data Base element. The number of values to be stored is determined when dimensioning the Data Base.

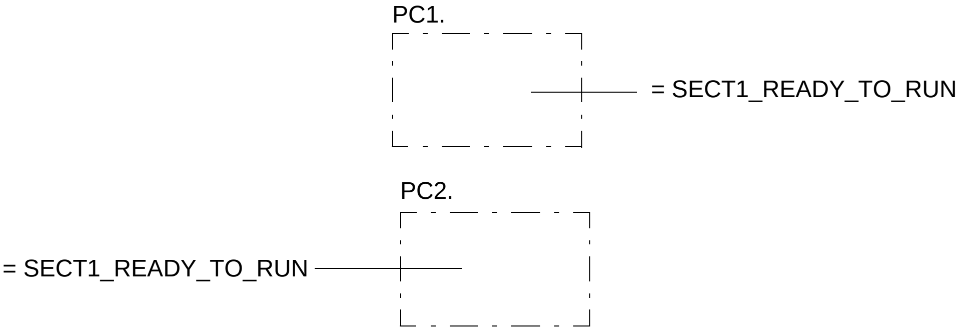


Figure 13-27. Connection between Different PC Programs

The storage files for data exchange between PC programs is determined by DAT elements. These may be used for different data types. The following is indicated when the Data Base element is created:

DAT(B)	for Boolean values
DAT(I)	for integers
DAT(IL)	for long integers
DAT(R)	for real numbers

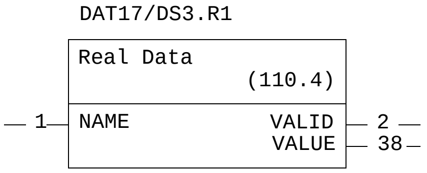


Figure 13-28. Data Base Element DAT(R)

Each DAT element for Boolean values holds 32 signals. For the other data types one Data Base element per value is used. Entering data to DAT elements is performed as for other Data Base elements.

DAT(B) is shown in Figure 13-29 and Figure 13-30. If an individual name for each Boolean value is desired only one value can be stored for each element. This value is then found in VALUE and is referred to with = DAT2 or = “the value at terminal NAME”.The other values are found by =DAT2:VALUEx or “Name”:VALUEx.

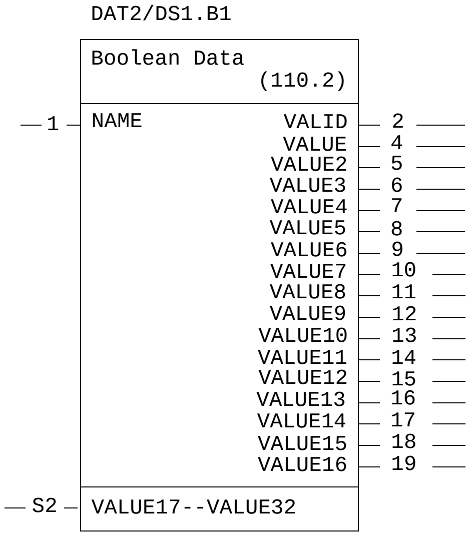


Figure 13-29. Basic Part of DAT(B)

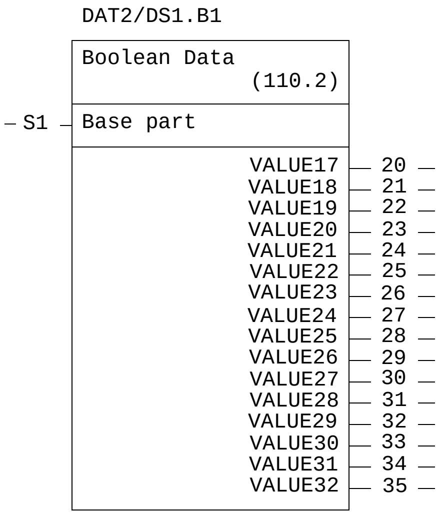


Figure 13-30. Values 17 .. 32 of DAT(B)

13.4 Preparing PC Diagrams

There are no rigid rules for the lay-out of a PC diagram. For ease of understanding, however, the PC elements should be arranged in order of execution. The automatic diagram printout places the PC elements mainly in this order. Space should be reserved for possible insertions.

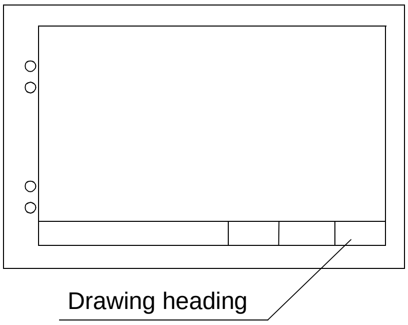


Figure 13-31. Diagram Form

As a recommendation, a few rules should be observed when preparing the draft PC diagram:

- Draw the draft on a horizontal A3 form, 420 x 297 m.m.
- Place drawing heading at the lower margin.
- Arrange the lay-out in the following way, see Figure 13-32:
  1. Use a left margin with space for references to other diagram pages.
  2. Use the center column for PC elements, symbolic names and connecting lines.
  3. Use a right margin with room for references to other diagram pages.

- The signal direction is always from left to right and top down.
- Place the PC elements starting in the top left hand corner with main direction diagonally downwards.
- Write the item designation of the PC module (the last term) above its top border.
- Write the superior item designation at the top of the diagram page.
- If the PC module occupies several pages, the complete item designation should also be written under the lower border.

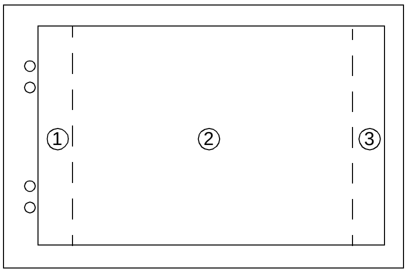


Figure 13-32. Lay-out of Diagram Form

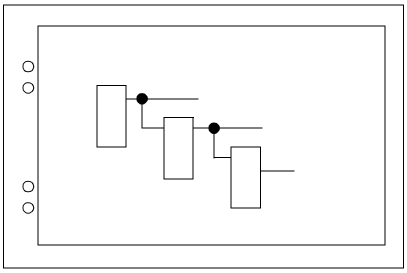


Figure 13-33. Main Direction of PC Diagram

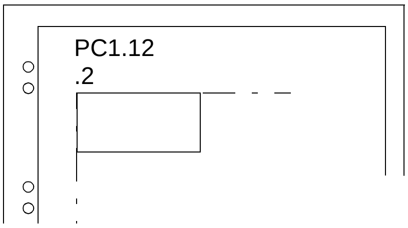


Figure 13-34. Placement of Item Designations

13.4.1 Page Numbering

The page numbering is determined at program entry using special commands.

Page numbers can be given in the range 1- 9999. Each page may optionally have sub-pages in the range 1 - 9. This is particularly useful when insertions have been made.

Examples: 10.3 7839.2

The Engineering Station can renumber the pages for example for removal of sub-pages and insertion of spare pages.

13.4.2 Printout of PC Diagram

With a command from the Engineering Station one or several pages may at a time be printed out in graphical format with selectable page sizes.

Common to all pages even a lot of items about customer, plant name, date and so on will be printed out. These items may be different for all PC programs.

The printout function will then take care of layout, cross references, signal names and so forth completely automatic.

Concerning the cross references these are for inputs against database or a page number from where the signal originates.

For outputs these are given to the database or all pages where the signal is used.

Chapter 14 Execution

The internal software of the Advant Controller 400 Series system is schematically illustrated by Figure 14-1.

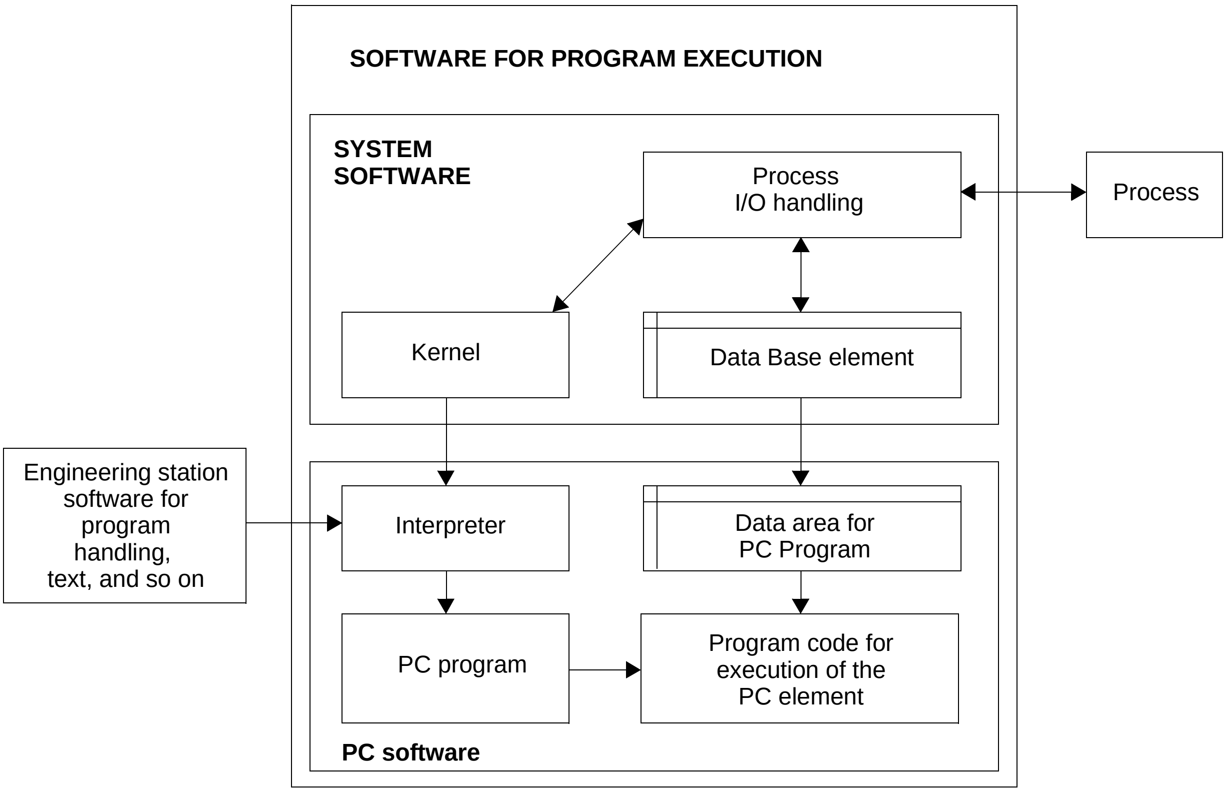

SOFTWARE FOR PROGRAM EXECUTION
SYSTEM SOFTWARE
Process I/O handling
Process
Kernel
Data Base element
Engineering station software for program handling, text, and so on
Interpreter
Data area for PC Program
PC program
Program code for execution of the PC element
PC software

Figure 14-1. Schematic Block Diagram of the Internal Advant Controller 400 Software

14.1 PC Programs

In the preceding chapters the block marked PC program has been described. This block consists of one or more PC programs with their execution units.

The execution units are:

- Control module
- Master- slave
- Sequence

To each PC program belongs a data area for storage of variable values of the PC element outputs, operational parameters and internal calculation values for the PC elements.

14.2 Interpreter

A special program called interpreter is internally organizing the execution of a number of execution units. This program must arrange for each execution unit to be executed with its requested cyclicity. Furthermore, special action must be taken by the interpreter if the execution unit is blocked, reset and so forth.

14.3 Kernel

Since the PC programs in a Advant Controller 400 Series system must co-exist with other types of programs such as Operator Station and communication programs the need arises to schedule the execution of the different programs. This task is performed by the so called kernel of the system software. For PC this involves starting the interpreter at the right times. If, for example, the shortest possible cycle time for a particular Advant controller 400 Series system is 10 millisecond, the interpreter will be started by the kernel every 10 millisecond.

The kernel will, at each point in time, schedule for execution the program with the highest priority. This may have the effect that the execution of an execution unit is interrupted temporarily by the reception of data from another system via a communications port or by the change of the value of an input on a digital input board. This however, will have no effect on the function of the PC program.

14.4 Process I/O Handling

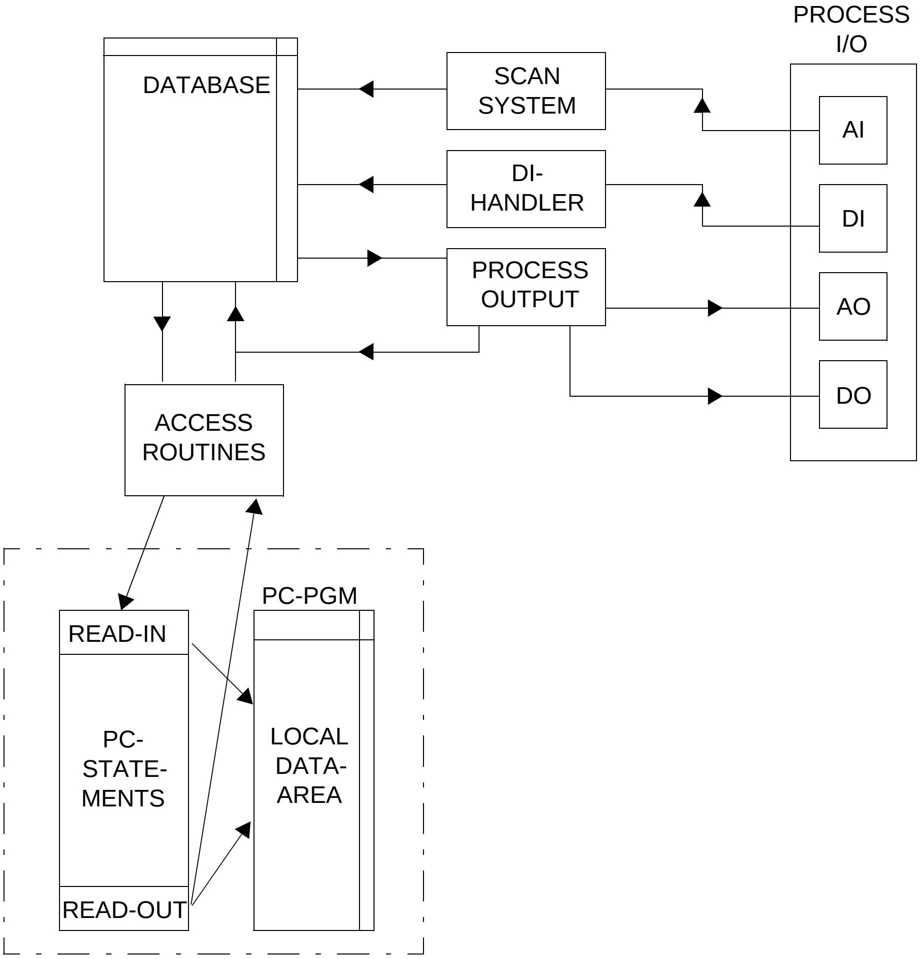


Figure 14-2. Process I/O - Data Base element - PC Program

Each Advant Controller 400 Series system has data exchange with the process. This is performed directly by bus connected I/O-units.

Current data for these units are stored in Data Base elements. This data contains information relevant to each board, for example board address, status and error signals. For each channel on the board there is information about the value of the signal, if the channel is blocked and so forth. The information normally used by the PC program is the current value of the signal.

Data exchange between the process and the Data Base element is performed by the process communications programs. For the most important types this is performed as follows:

DI:	Each digital input board interrupts the processor when an input signal changes value. When this occurs the kernel immediately starts the program for storage of the new value in the Data Base element. The DI signals may instead be cyclically scanned. This is selected with a parameter for the Data Base elements.
AI:	Each channel of an analog input board is read cyclically. The cycle time, being stored in the Data Base element, can be selected individually. The kernel starts the program for storage of the value of an analog inputs in the same way it starts the interpreter. The value read is filtered, rescaled to engineering values, limit checked and so on.
DO:	Each time an execution unit in a PC program is executed, the output values are stored in the Data Base element. If a change has occurred since the previous execution an output program is started for output to the board.
AO:	Analog outputs are handled in the same way as digital outputs. A conversion from engineering values to suitable values for the board is performed prior to output.

14.5 Cycle Time Table

To the interpreter belongs a number of cycle time tables. These indicate which execution units have a particular cycle time as illustrated by Figure 14-3.

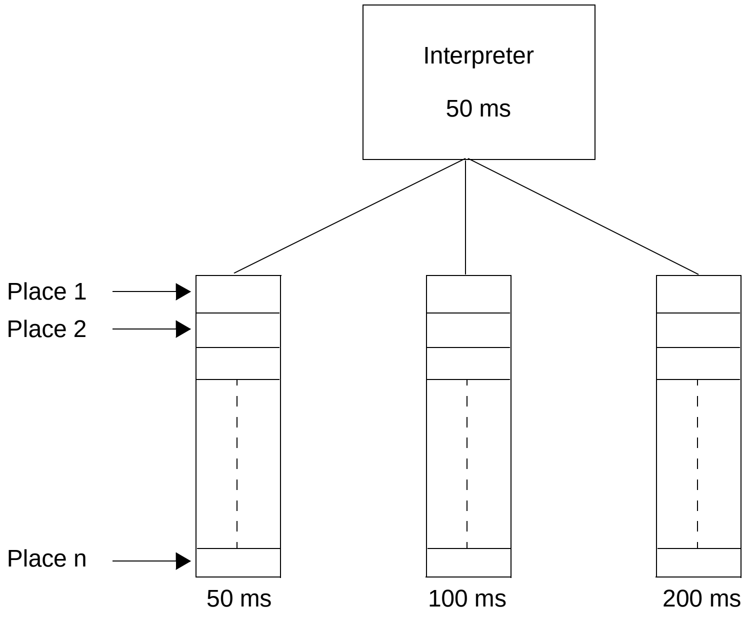


Figure 14-3. Cycle Time Tables, Example

When a cycle time is selected for a certain execution unit an internal reference to the unit is stored in the corresponding cycle time table.

Using the call parameter “place in cycle time table” the execution unit may be assigned a certain place in this table. In this manner the execution order between units with the same cycle time is determined.

The execution units in the cycle time tables are grouped internally. In the example in Figure 14-3 this has the effect that every 50 milliseconds all units in the 50 milliseconds table are executed along with half the units in the table for 100 milliseconds and a fourth of the units in the 200 milliseconds table Through this arrangement the system load is well distributed over time.

14.6 Several Interpreters

In a system where the difference between the shortest and longest cycle time is large, for example 10 milliseconds and 2 seconds, an internal prioritization must be performed.

Assume that an execution unit in the 2 seconds table takes 20 milliseconds to execute. This unit would then every 2 seconds affect the starting time for the units in the 10 milliseconds table.

For this reason the Advant Controller 400 Series systems have 3 interpreters called A, B and C, of which A has the highest priority, see the example in Figure 14-4.

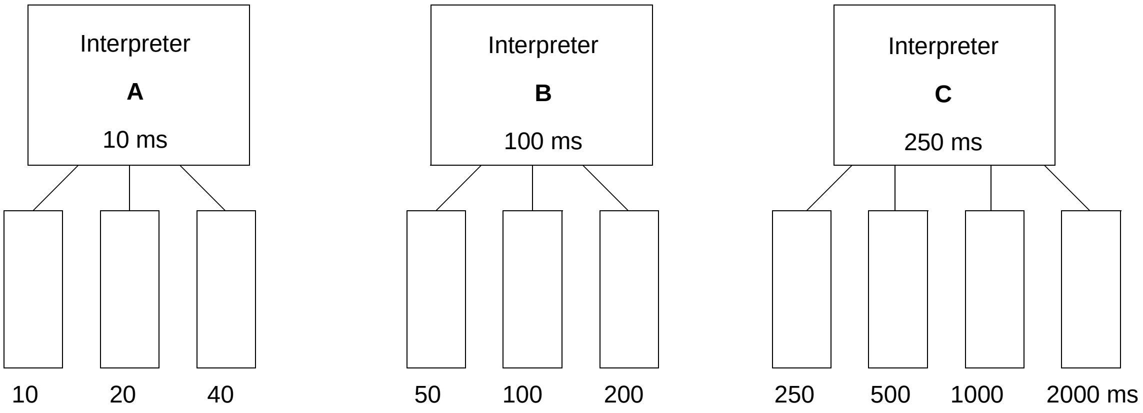


Figure 14-4. Interpreters A, B and C

Execution under interpreter C may be interrupted by A and B, execution under B may be interrupted by A which in turn is not interruptible.

14.7 Input and Output Phases

To isolate each execution unit from data change during execution the execution is commenced by an input phase and ends with an output phase.

14.7.1 Input

During the input phase the values in the Data Base elements are stored in the local data area. In addition, data from other execution units belonging to an interpreter with higher priority are stored at special locations in the local data area.

The necessity to have corresponding data during the entire execution of a unit may be illustrated by Figure 14-5.

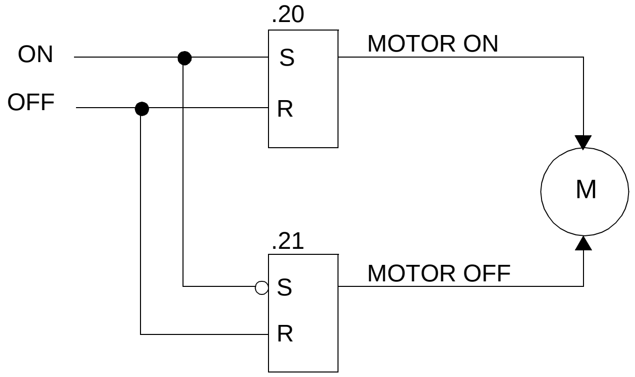


Figure 14-5. Example of the Necessity of Having Input Elements

If an internal interrupt was to occur between the execution of PC element 20 and 21 in the figure because the input ON changes value the motor would be ordered on and off simultaneously.

14.7.2 Output

All used data from all PC elements in an execution unit are stored in the Data Base elements at the same time, during the output phase. Data is fetched directly from the storage location of each PC element output.

Data from execution units belonging to a certain interpreter are, if connections to PC elements under execution units with higher priority exist, stored in special locations in the local data area during the output phase. This is necessary to get corresponding data items from execution units with lower priority.

Data that belong to an output or input are defined by so called output or input elements respectively. These are automatically inserted by the tool and need not normally be handled. In certain test situations, however, they may have some significance and they are therefore indicated in the documentation with a G (get) and a P (put) in the PC diagram. The need for this can be seen from the example in Figure 14-6.

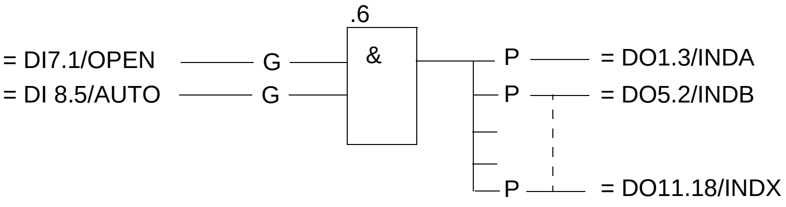


Figure 14-6. Input and Output Elements

If the PC element .6 belongs to a blocked module the data read at the PC element output may be different from that stored in the Data Base element. This may also be the case for a module belonging to an execution unit under a different interpreter. The indication in the PC diagram is made to avoid possible misunderstanding.

Figure 14-7 summarizes how data exchange between PC elements and Data Base elements and between PC elements in different execution units under different interpreters is performed.

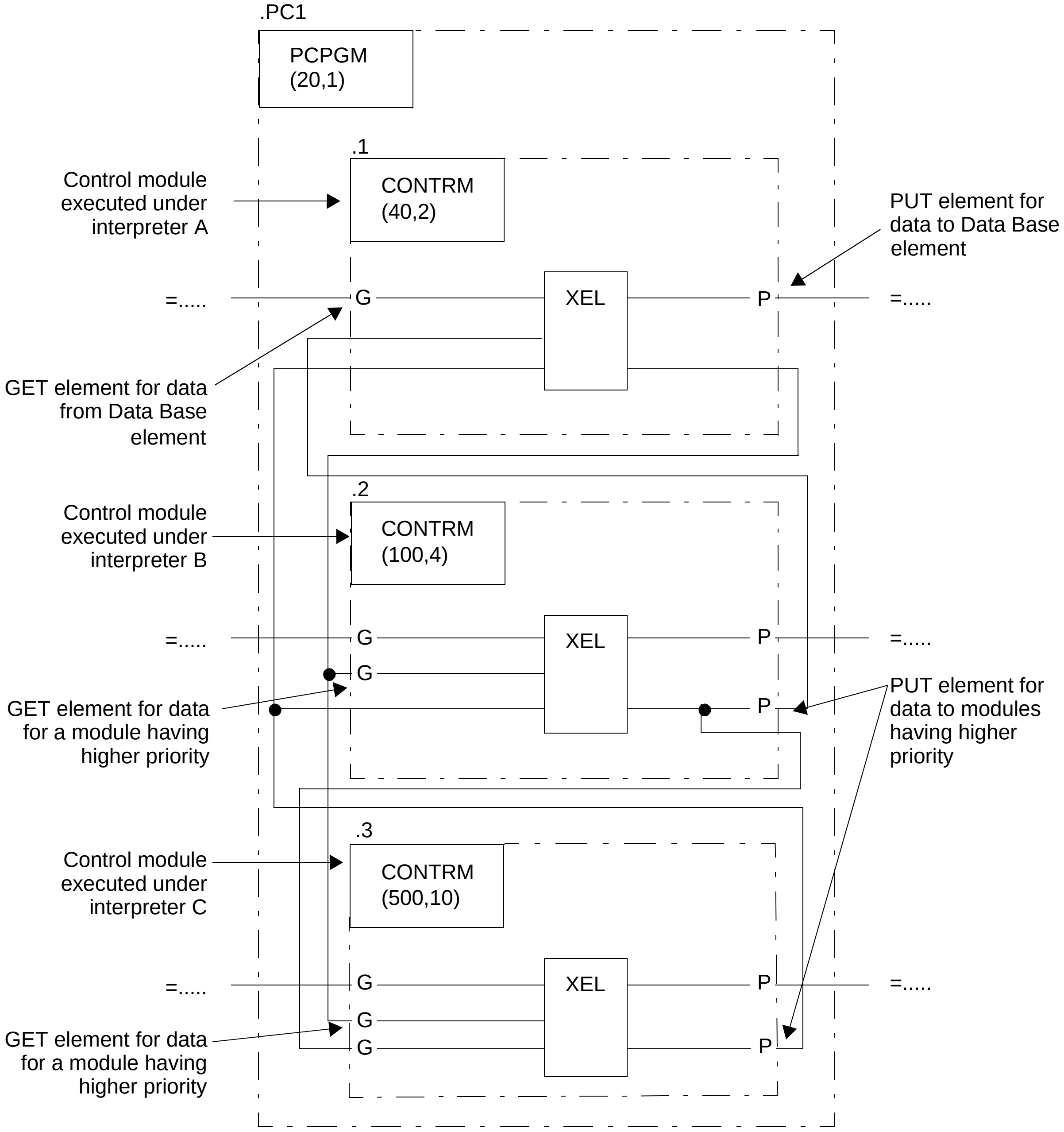


Figure 14-7. Data Transports

14.8 Execution Order

The following order of execution apply for the different execution units in a Advant Controller 400 Series system:

Control Module

1. Structure element CONTRM
2. Input elements in the control module
3. PC elements in the control module
4. Output elements in the control module including those for the CONTRM

Master-Slave

1. Structure element MASTER
2. Output elements from MASTER
3. Input elements of the first slave
4. PC elements in the first slave
5. Output elements of the first slave
6. Input elements of the second slave
7. PC elements in the second slave
8. Output elements of the second slave

.

.

.

Output elements of the last slave

Sequence

1. Structure element SEQ
2. Structure element STEP
3. Input elements of the active step
4. PC elements of the active step
5. Output elements of the active step including those for the step header
6. Output elements of SEQ

On step change the active step execution is repeated and the new step is then executed. This is always performed for the entire step with its input and output elements.

PC Program

1. Structure element PCPGM
2. Output elements from PCPGM

14.9 Execution of PC Elements

Following execution of the header of an execution unit and the execution of the input elements execution of the PC elements commences. This execution is controlled by the interpreter by calling the program code in the PC element library for each PC element. This is done in order of storage which also corresponds to the entry and documentation orders.

The program code in the library works with a part of the PC program corresponding to a single PC element. The internal working principle is shown below for an ANDgate with 2 inputs.

The identity of the PC element (5 in Figure 14-8) is only used by the Engineering Station for example when changing, testing and generating documentation. The AND part is used by the interpreter to select the appropriate code from the library.

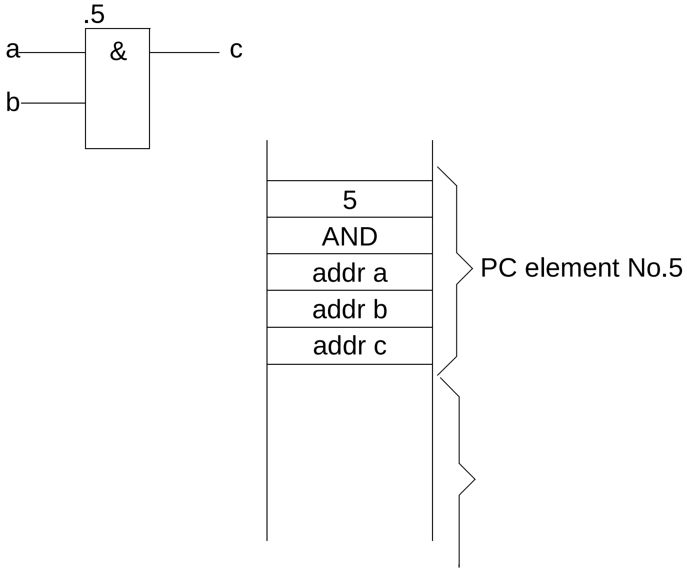


Figure 14-8. Principle for Internal PC Element Storage

This program code reads data from the data area at addresses a and b and calculates the logical AND expression of this data. The result is then stored at address c. The interpreter now regains control and proceeds with the next PC element in the PC program.

14.10 Reset Execution

Clearing or reset execution takes place when an R input of a module header is active. This has the effect of setting all variables in the local data area of the module to an initial value. For most variables this implies that they are set to the zero value of their respective data type. There are, however, certain exceptions from this, since the outputs of a certain PC element must have consistent values, see Figure 14-9.

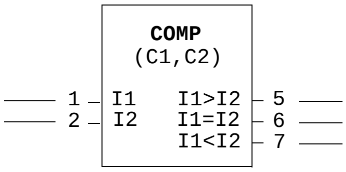


Figure 14-9. Comparator COMP-I

Since X and Y both are 0 with reset execution the output I1 = I2 must be 1 and the other outputs be 0 for the outputs to have consistent values.

Following reset execution output to the Data Base element is performed with the values generated in the local data area.

The following apply to the different execution units when R = 1.

Advant Controller 400 Series

CONTRM	All PC elements each cycle
MASTER	The elements of all slaves each cycle
SEQ	All PC elements of all steps one cycle when R goes to 1
PCPGM	All constituent execution units according to the above

14.11 Initialization

The data in a Advant Controller 400 Series system must be assigned defined values following power-up. This is performed by executing all modules and PC elements in so called init mode. To the PC elements this is normally equivalent with reset execution. Possible exceptions from this are stated in the *PC Elements Advant Controller 400 Series Reference Manual*.

Data Base Elements

As opposed to the case at reset execution no read-out to the Data Base elements takes place on initialization. This updating is done at the first normal execution cycle.

This makes it possible to retain values in the Data Base element across a power failure. The PC program must be designed to handle data correctly after the power failure. This is mainly applicable for data between PC elements. These data are stored in DAT elements.

Chapter 15 Applications with Sequences

The function of the sequence and step modules are described in Chapter 10 Sequence and Step Modules. Construction of straight sequences and sequences with branches and reunions is described below.

15.1 Function Diagram

Symbols according to the IEC Standard 1131-3 were used in the figures in Chapter 10.

Such a diagram should always be prepared when constructing control programs with sequences. This will provide the structure and main functions of the control in an easily surveyed document.

According to the IEC Standard 1131-3 each step in the sequence is indicated by a square in which the number of the step is written, see Figure 15-1.

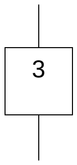


Figure 15-1. Step

An initialization step is indicated with a double square as in Figure 15-2.

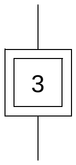


Figure 15-2. Initialisation Step

The condition for step change (transition condition) is indicated with a line across the connecting line between two steps. In Figure 15-3 the transition condition is that the variable a is true, that is step change is performed if step 5 is active and the variable a is true.

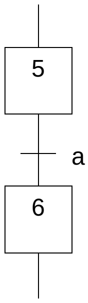


Figure 15-3. Transition

There must always be a transition condition (a simple variable or an expression) between two steps.

This condition may be stated either in plain language or in Boolean form as shown in Figure 15-4.

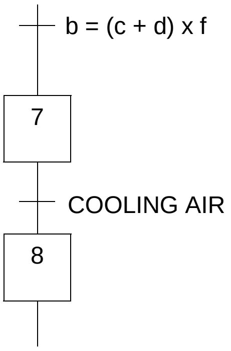


Figure 15-4. Example of Transition Conditions

There must be only one condition between two steps.

Figure 15-5 shows a comparison between the function diagram presentation and the corresponding circuit, in logic diagram form.

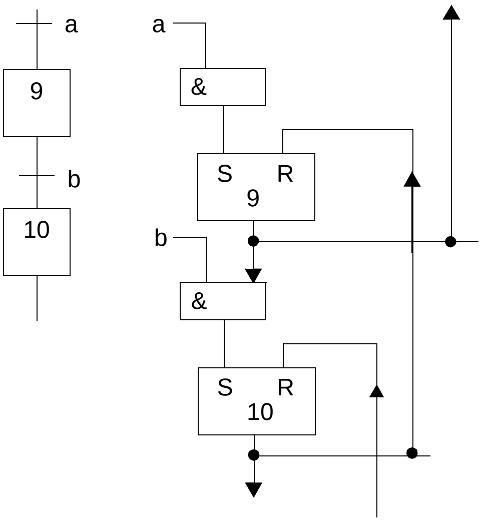


Figure 15-5. Function Diagram and Logic Diagram

15.2 Activities

The activity in each step is written in a separate box connected to the step square as shown in Figure 15-6. If several activities are to be performed in the same step, each activity is allocated its own box as shown in Figure 15-7.

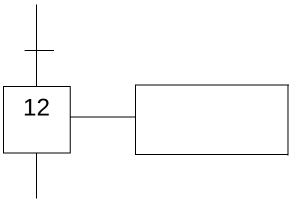


Figure 15-6. Activity Box

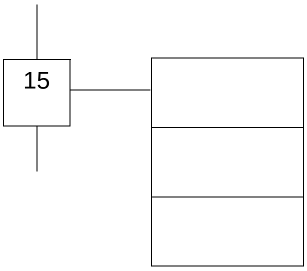


Figure 15-7. Step with 3 Activities

The activity box may be divided into three fields, designated a, b and c in Figure 15-8.

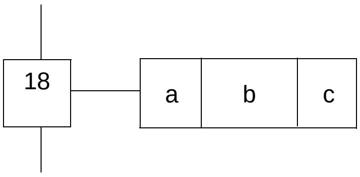


Figure 15-8. Step with Description Fields

a. Type of activity denoted by a letter:
   C Continuous order as long as the step is active
   T Time controlled order
   S Activity with memory function, that is the activity is to continue even after the step has been exited. (S = Stored).

b. Activity including any additional conditions

c. A reference number if several activities occur in the step. This field is left out if no references are needed.

15.2.1 Continuous Order (C)

A C-order means that the activity is to continue as long as the step in question remains active. The activity may also be combined with different conditions so that the order is active only when the condition is true.

In Advant Controller 400 Series systems, the RUN-output of the step module header is 1 while the step is active. This output may be used for the C- activity. In the example of the car sequence in Chapter 10, the RUN output may be connected directly to the physical outputs for motor forward/reverse via the Data Base element.

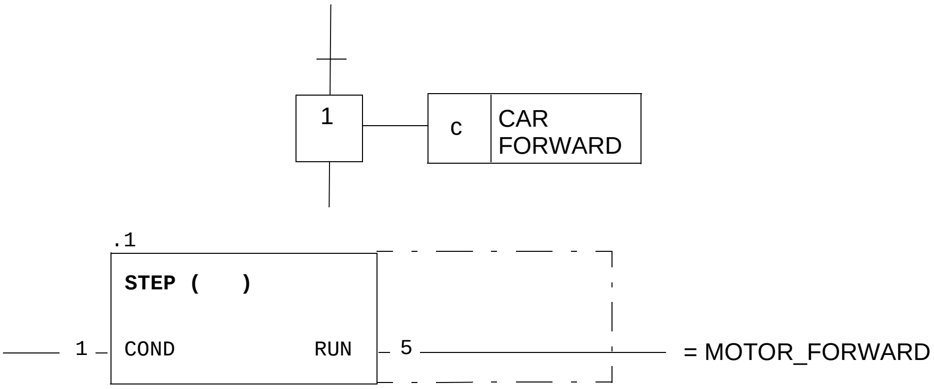


Figure 15-9. Example of C-order

Any additional conditions, for example that a hatch in the bottom of the car must be closed for the car to be run is implemented by introducing an AND gate in the step module as shown in Figure 15-10.

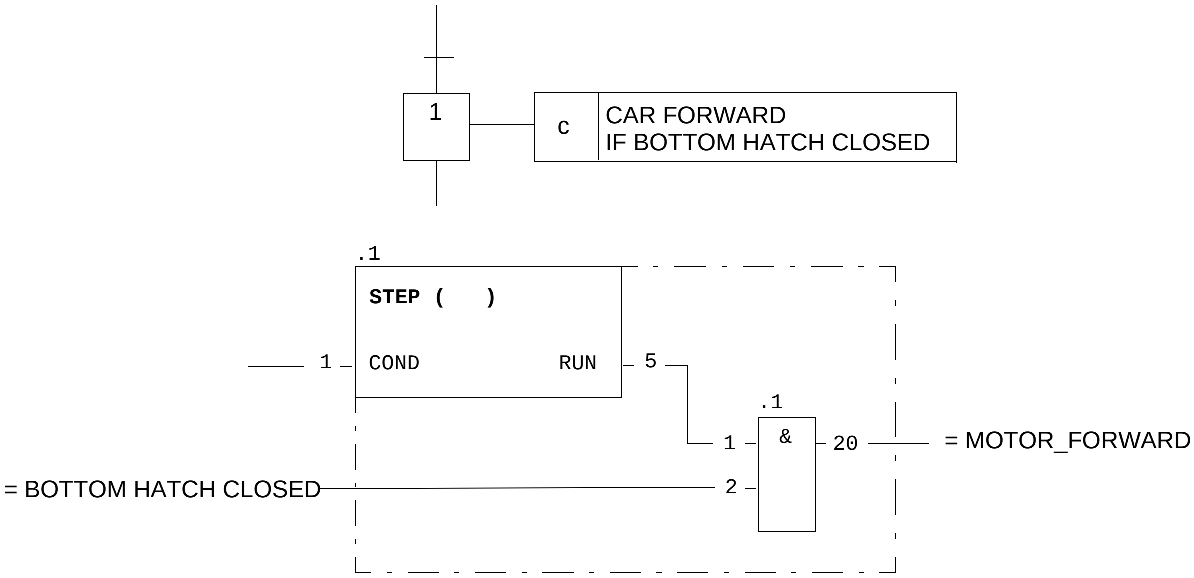


Figure 15-10. Example of C-order with Condition

15.2.2 Time Controlled Order (T)

An activity may be delayed or time limited. Such time functions are specified with the letter T and are identified with the number of the step.

Delayed Activity

Figure 15-11 shows a delayed activity: A valve is to be opened 10 seconds after the activation of step 6. Figure 15-12 shows the controller implementation.

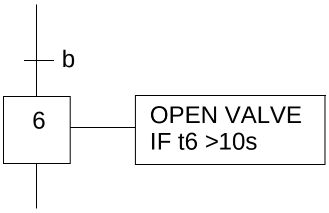


Figure 15-11. Delayed Activity

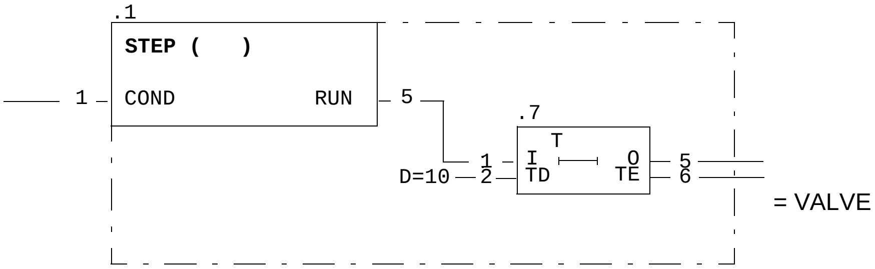


Figure 15-12. Example of Delayed Activity

The internal delay circuit in the step module header may also be used. See Figure 15-13.

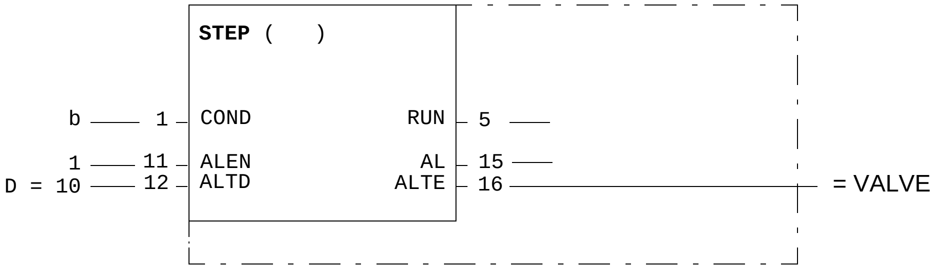


Figure 15-13. Example of the Use of the Time Circuit of the Step Module Header for Delayed Activity

Time Limited Activity

A time limited activity is specified in accordance with Figure 15-14. In the example, the activity “HEATING” is to start when step 14 is activated and is to continue for 25 minutes. It is assumed that the step is active at least as long as the duration of the activity.

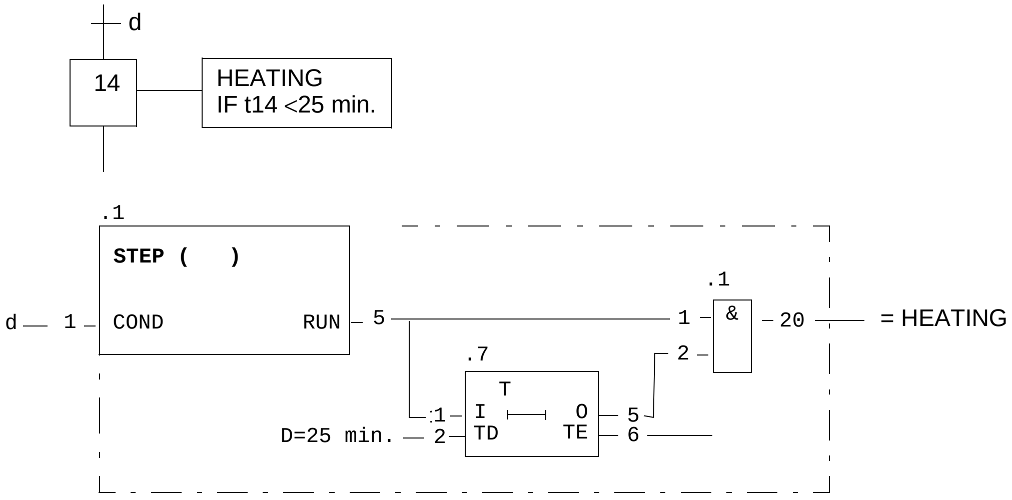


Figure 15-14. Example of a Time Limited Activity

A time limited activity that often occurs is that of a step that is to be active for a specified time. This is shown in Figure 15-15 which also shows an implementation.

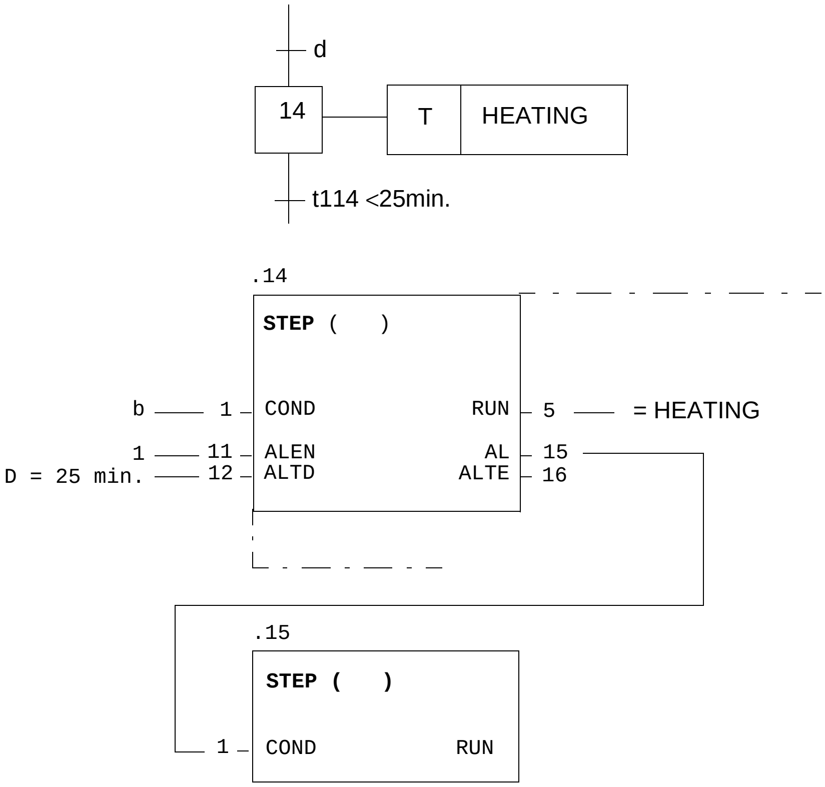


Figure 15-15. Example of a Step being Active for a Certain Time

15.2.3 Activities with Memory Function (S)

An S-activity is an activity that starts in a step and continues even after the step is exited. An example is a start-up sequence in which a number of activities are to be started in a specific sequence.

Another example is a specific activity to be started in one step and stopped a number of steps further on in the sequence.

No special measures need be taken to implement an S-activity in a Advant Controller 400 Series system. The execution of a step ceases when it is left and the data calculated in the step remains unchanged until the step is reset or re-executed.

An S-activity may also be implemented by activating a Data Base element in one step and later, in another step, deactivating that variable as shown in Figure 15-16.

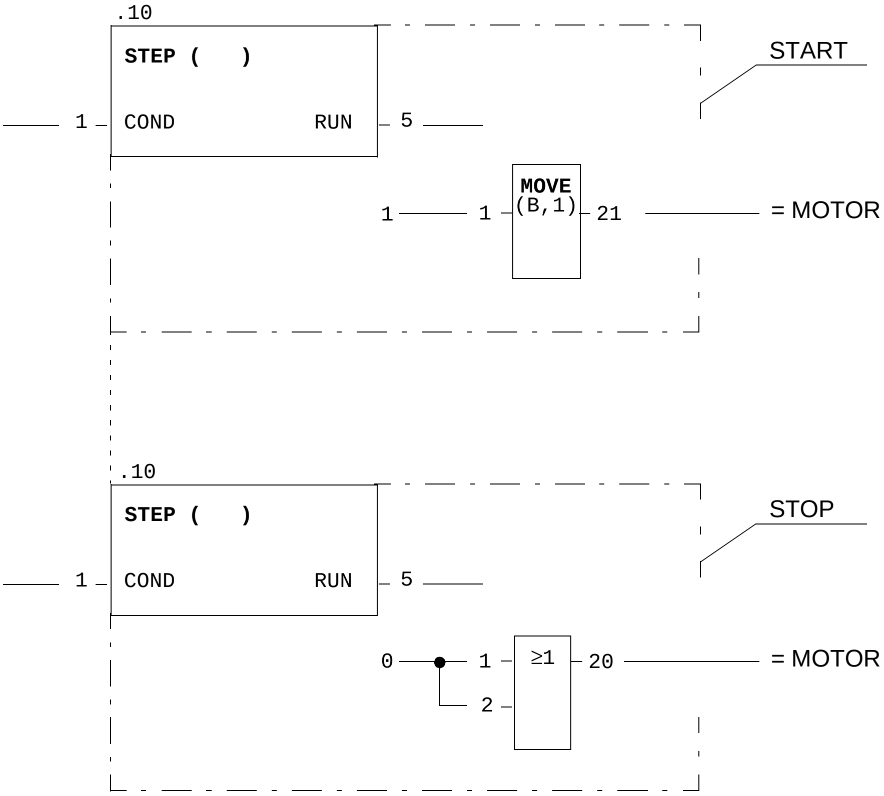


Figure 15-16. Example of Activity with Memory Function

A motor is started in step 10 by setting the Data Base element MOTOR to 1. When step 25 is activated the motor is stopped by setting MOTOR to 0.

Another implementation of this example is shown in Figure 15-17. By using the programmable reset function of the step header step 10 is reset when step 25 is exited. This solution is suitable when several activities are to be reset simultaneously.

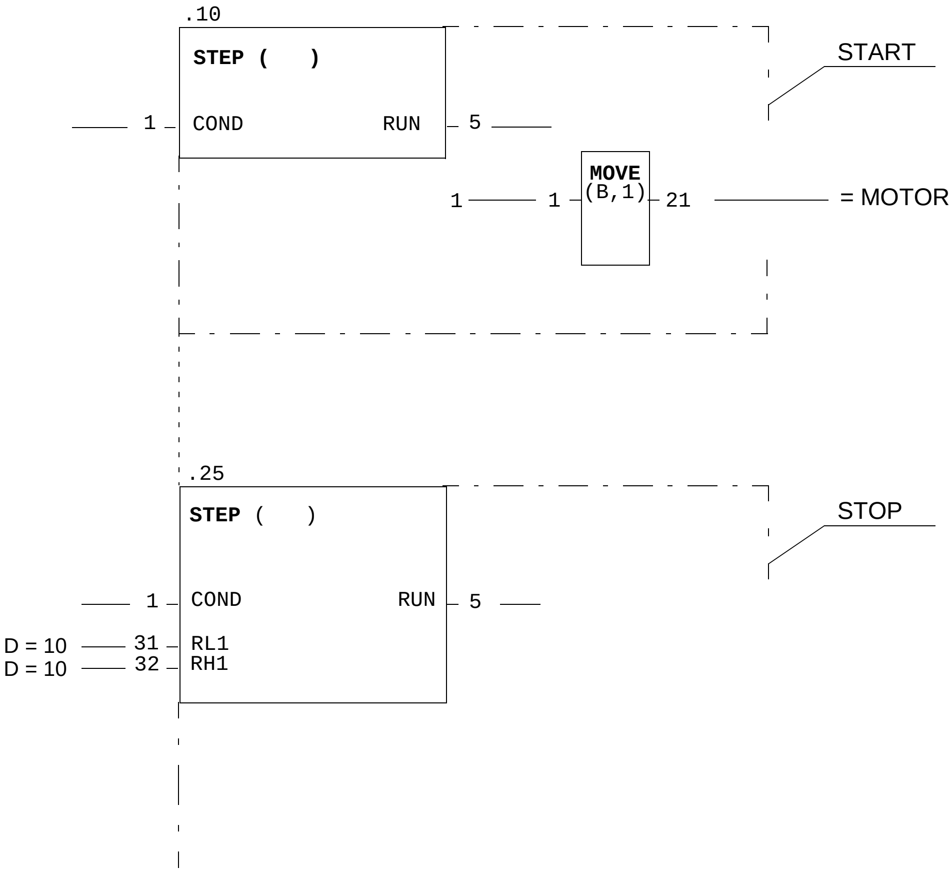


Figure 15-17. Example of Activity with Memory Function

Resetting a sequence implies resetting all steps in it. This must be considered if a Data Base element is to be active after the sequence is reset.

15.3 Single Sequences

A single sequence is made up of a series of steps which will be activated one after another.

Start

The sequence is started by activating the START (or STARTC) input and is reset by activating the R input.

To activate step 1 in Figure 15-18 the start condition a of the step must be satisfied. If the COND input is connected to a constant 1 the step will be automatically activated when the sequence is started.

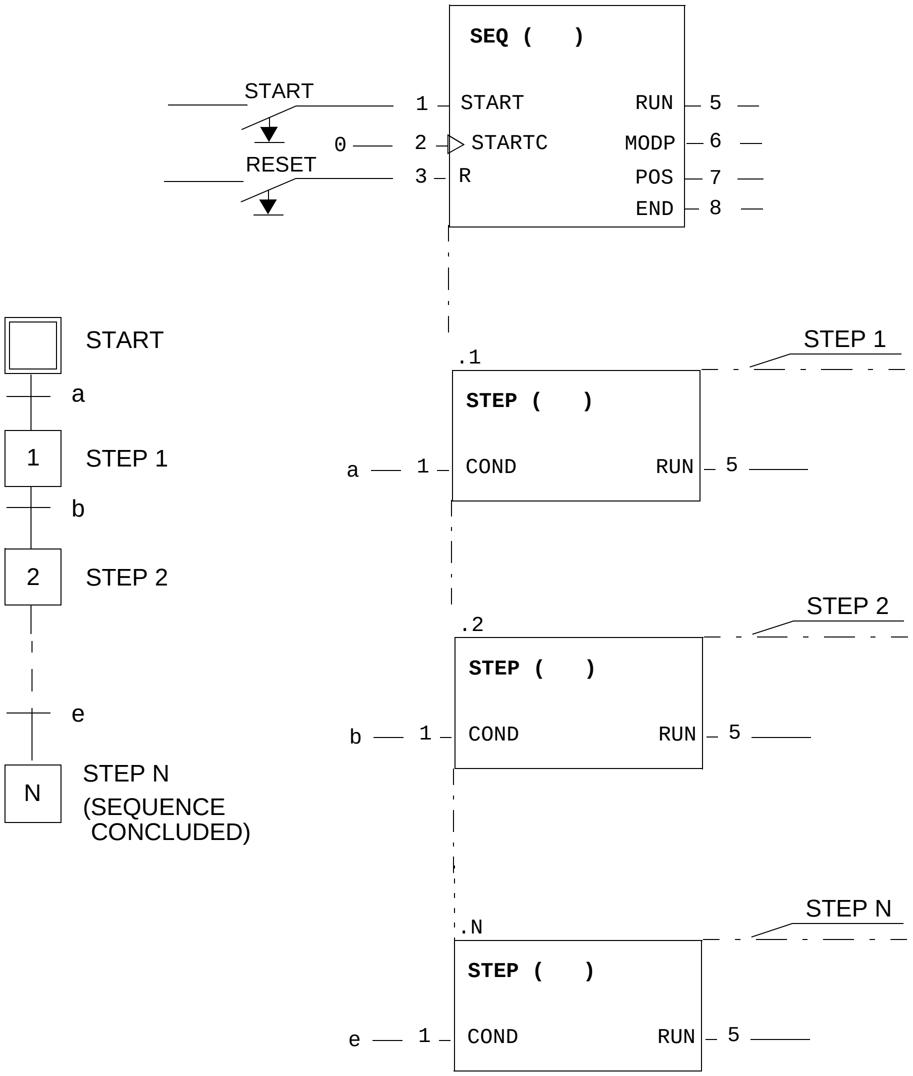


Figure 15-18. Example of a Single Sequence

Automatic Resetting

A sequence may be automatically reset when it reaches its last step if, for example the RUN output of the last step is connected to the R input of the sequence header. The same effect is achieved by connecting the END output of the sequence header to the R input. END is active when the last step of the sequence is active.

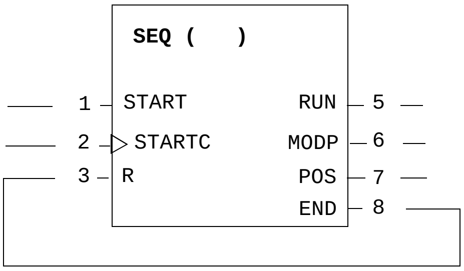


Figure 15-19. Automatic Resetting

Automatic Resetting with Restart

The sequence is automatically restarted if the START input is constantly held at logic 1.

In Figure 15-20 the sequence restarts as long as the switch AUTO is closed.

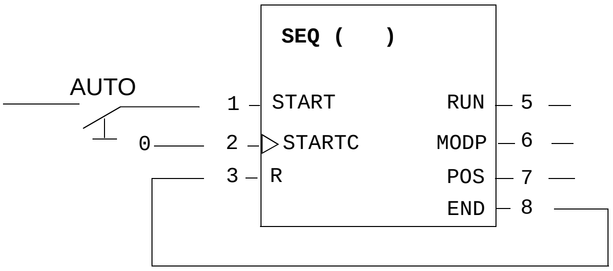


Figure 15-20. Automatic Resetting with Restart

Automatic Resetting without Restart

To avoid unwanted restarts caused by a start order remaining when the sequence is reset, the dynamic input STARTC can be used as in Figure 15-21.

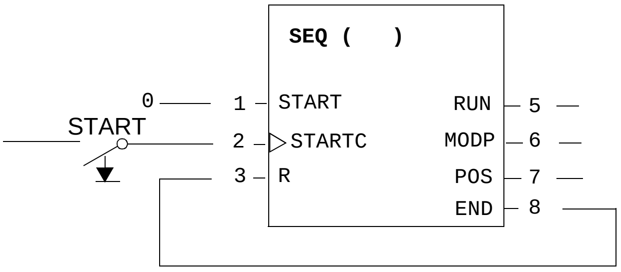


Figure 15-21. Automatic Resetting without Restart

15.4 Beginning of Sequence Selection (Divergence)

A sequence selection is a conditional choice between two or more branches. In the example in Figure 15-22 the sequence is to take the left branch if the condition **e** is satisfied and the right branch if the condition f is satisfied. If both conditions apply the right branch will be selected since the jump conditions are examined first. Refer to Section 10.6.5 Conditional Jump.

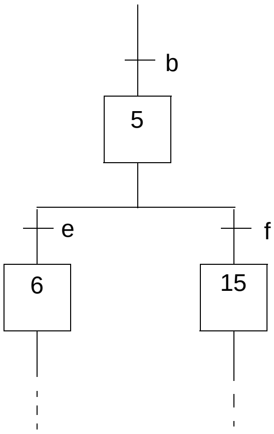


Figure 15-22. Sequence Selection

This type of structure may sometimes also be called divergence.

Figure 15-23 shows an implementation of this branch. The sequence may be seen as a single sequence with a side branch from step 5.

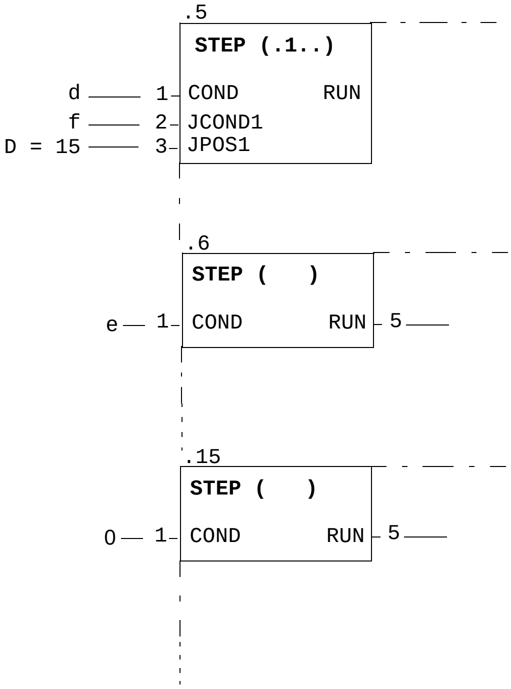


Figure 15-23. Branch in the Controller

For step 5 the call parameter C2 = 1, that is one jump function. The main branch is laid out in the normal way. The JCOND1 input of step 5 is connected to the jump condition **f** and the JPOS1 input is connected to the jump address D = 15.

NOTE

The COND input of step 15 must be set to 0 to prevent that an active step 14 does not cause an unintended transition to step 15.

The general form of a jump is described in Section 10.5 Sequence Header- SEQ. Up to four branches in addition to step change to the next step may be specified in each step of a sequence. Figure 15-24 and Figure 15-25 shows an example with a suggested implementation.

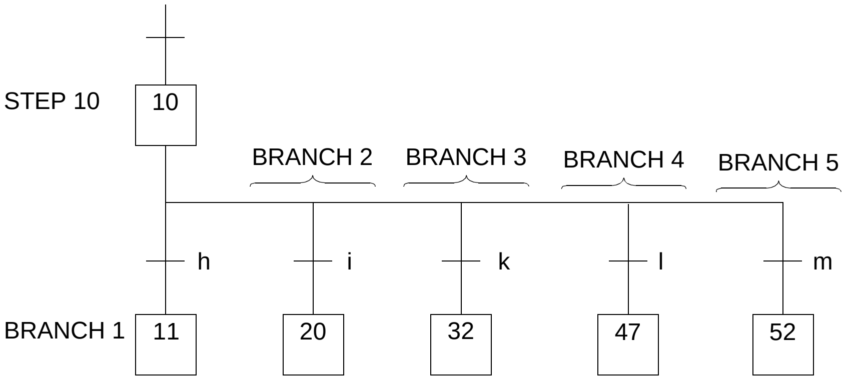


Figure 15-24. Branch

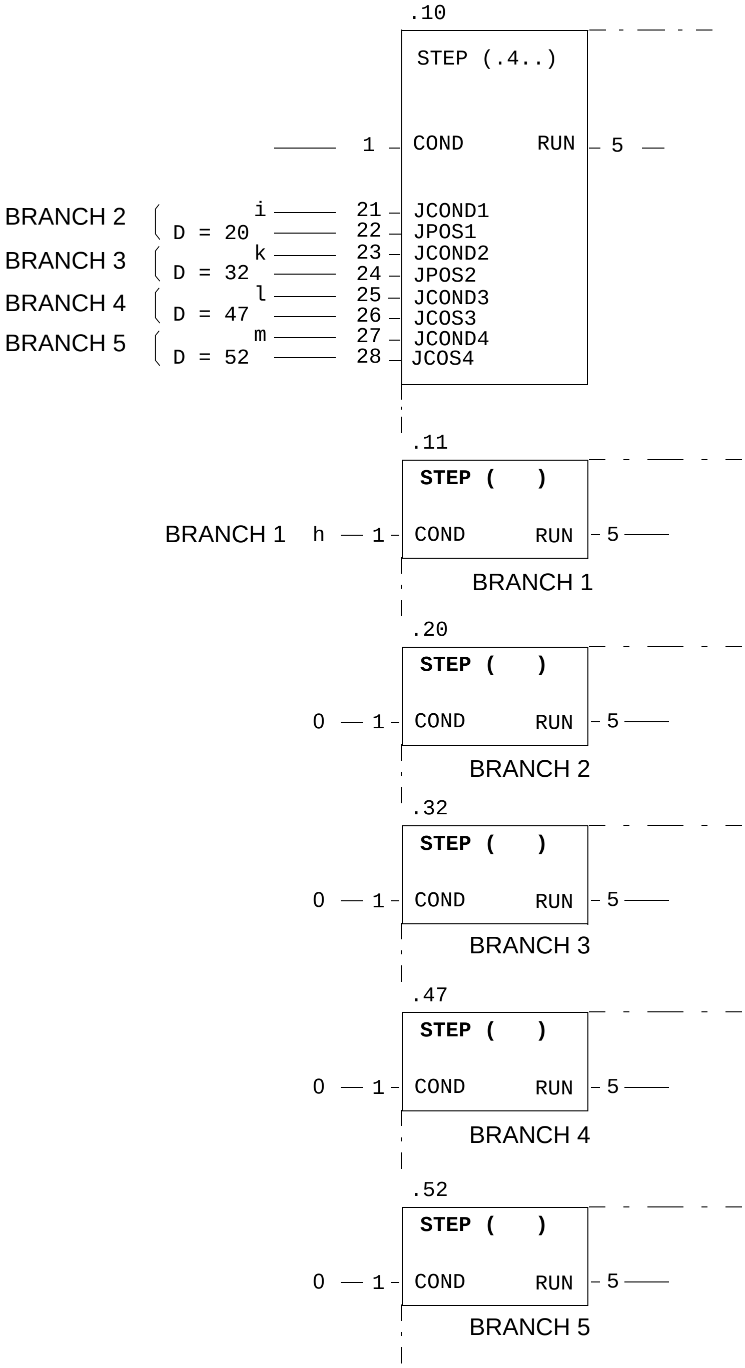


Figure 15-25. Implementation of the Branch in Figure 15-24

The jump addresses in the example are given as operational parameters (D = ...). They may also be given as constants (CD = ...).

A jump address may be calculated by the program in one of the steps of the sequence or in another program section.

A branch may be controlled by, for example, an operational selector in the control desk.

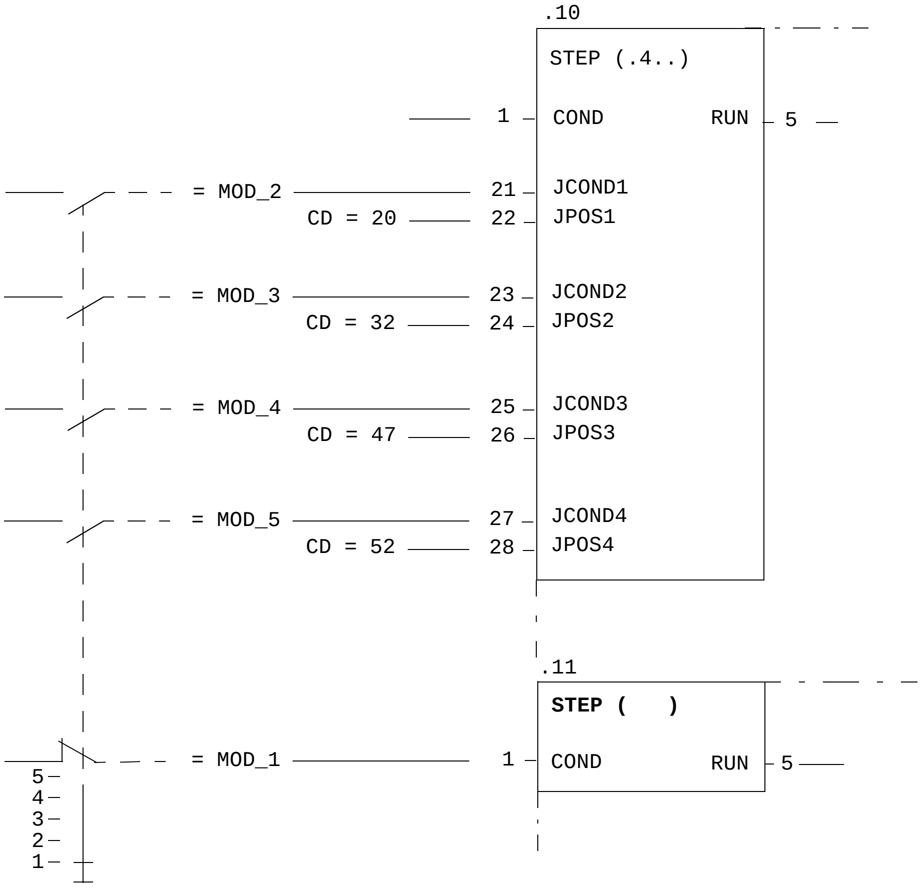


Figure 15-26. Branching Controlled by a Selector

15.5 End of Sequence Selection (Convergence)

The end of sequence selection of two or more alternative branches is implemented, in principle, in the same way as branching by means of jumps defined by JCOND and JPOS. This end is also often called convergence.

Figure 15-27 shows a sequence with two branches that are converged in step 21. Two different conditions for continuation into the common branch has been given in Figure 15-27. Nothing prevents the use of the same condition as shown in Figure 15-28.

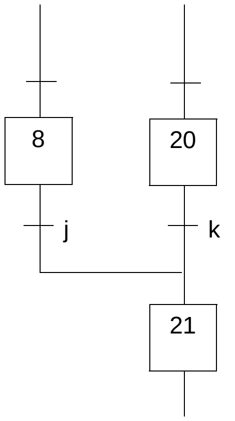


Figure 15-27. End of Sequence Selection, Different Conditions

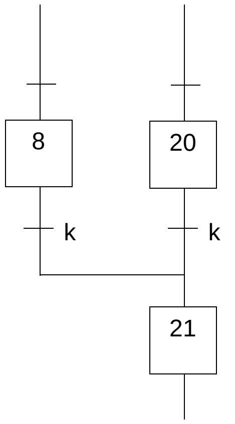


Figure 15-28. End of Sequence Selection, Equal Conditions

The convergence is implemented by giving a jump to step 21 from step 8 with the jump condition **j** as in Figure 15-29. The COND input of the step after step 8 (step 9) must be set to 0 to prevent an unintended step change from 8 to 9.

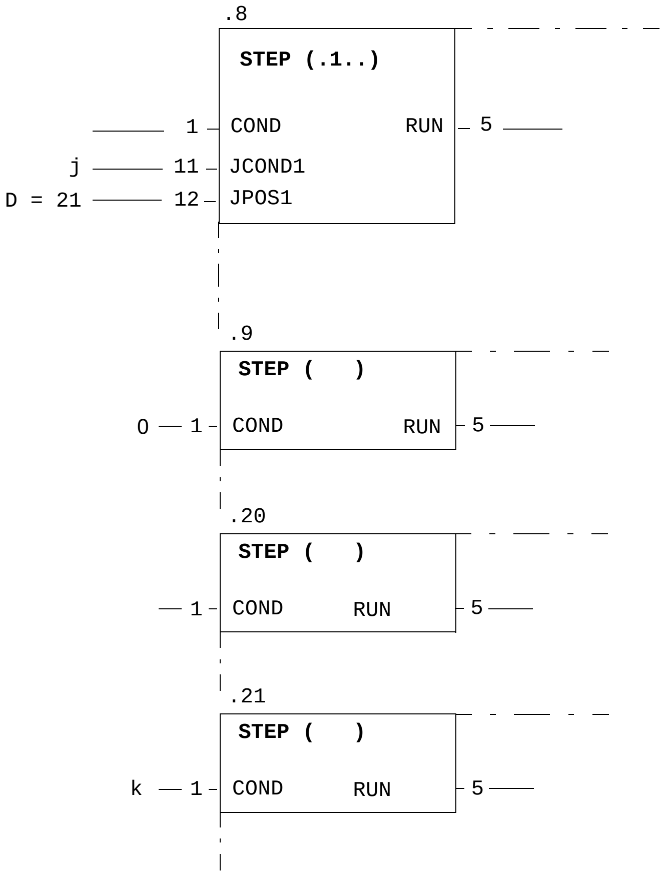


Figure 15-29. Implementation of the Convergence in Figure 15-28

15.6 Complete Sequence Selections

A sequence may have alternative branches for parts of the sequence. Figure 15-30 shows an example and the implementation with a combination of a branch and a convergence is shown in Figure 15-31.

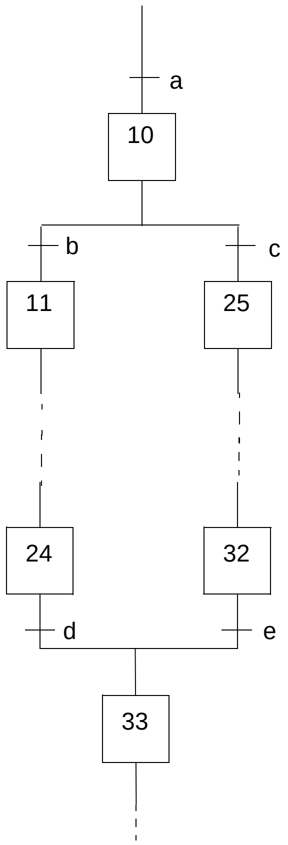


Figure 15-30. Combined Branch and Reunion

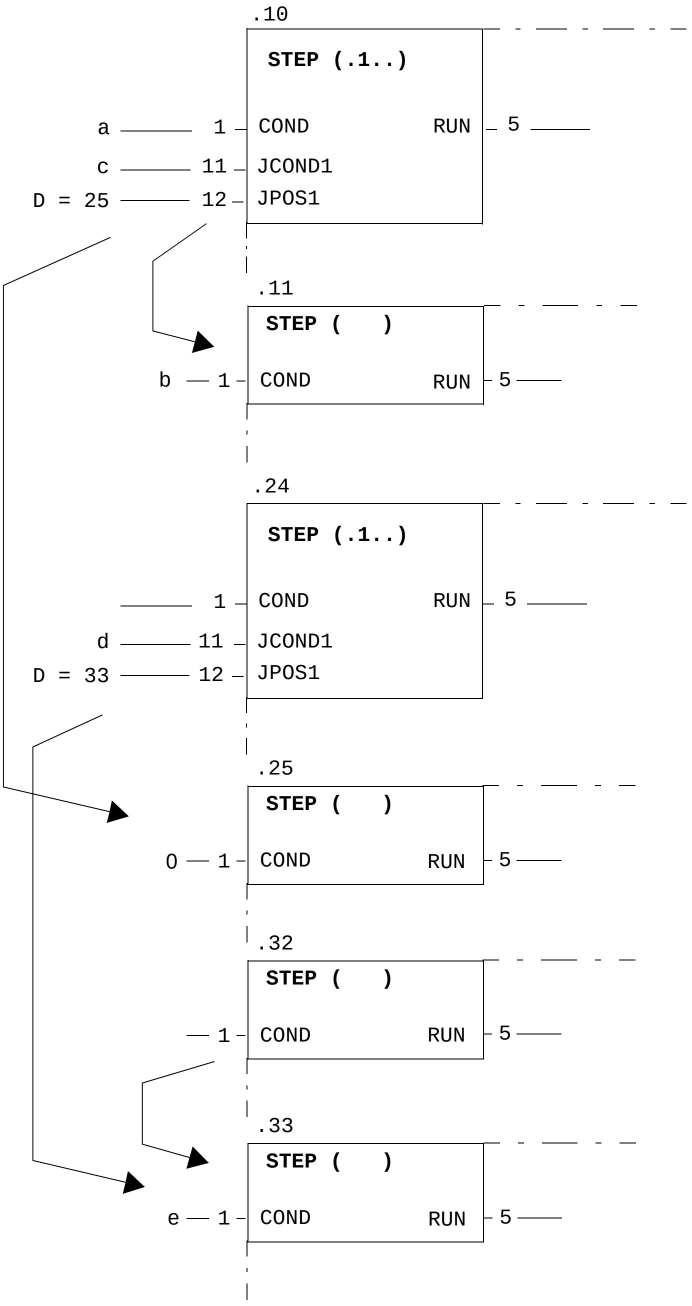


Figure 15-31. Implementation of a Combined Branch and Convergence according to Figure 15-30

15.7 Sequence Selection with Several Terminating Branches

In a sequence with several terminating branches as in Figure 15-32 each branch must finish with a jump to a last step common to all branches. This is necessary to obtain a single last step of the sequence.

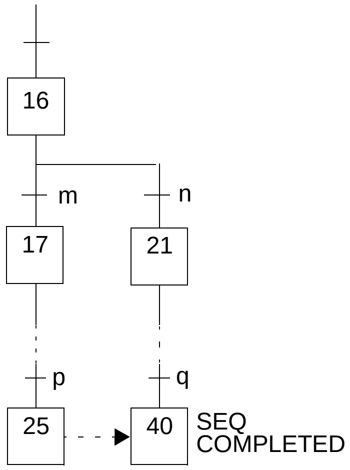


Figure 15-32. Branch with Terminal Branches

The implementation depends upon the layout of the particular sequence. A suggested solution is shown in Figure 15-33.

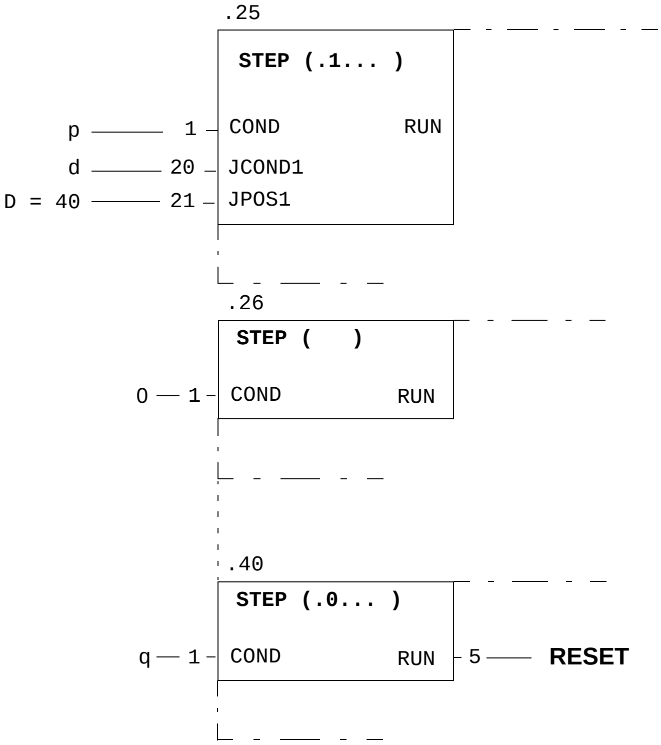


Figure 15-33. Implementation of a Sequence according to Figure 15-32

15.8 Simultaneous Sequences

If in a sequence there are to be simultaneously active branches, for example as shown in Figure 15-34, the side branches must be separate sequences.

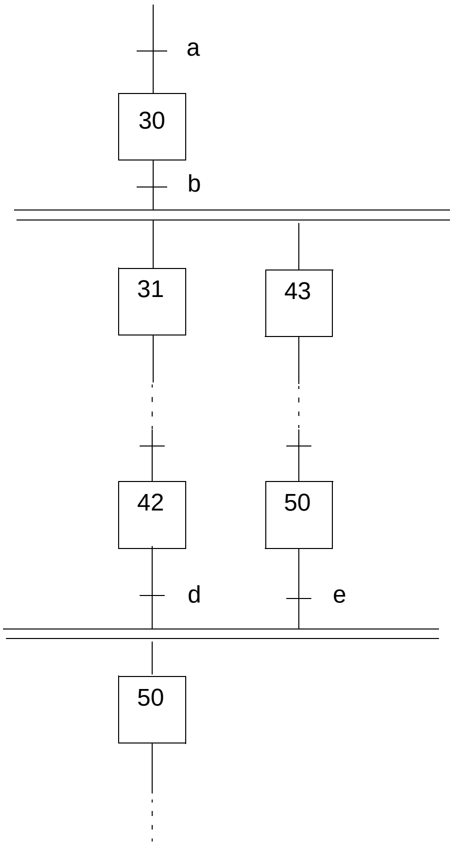


Figure 15-34. Simultaneous Sequences

Both branches must have to run to the convergence step before the main sequence proceeds.

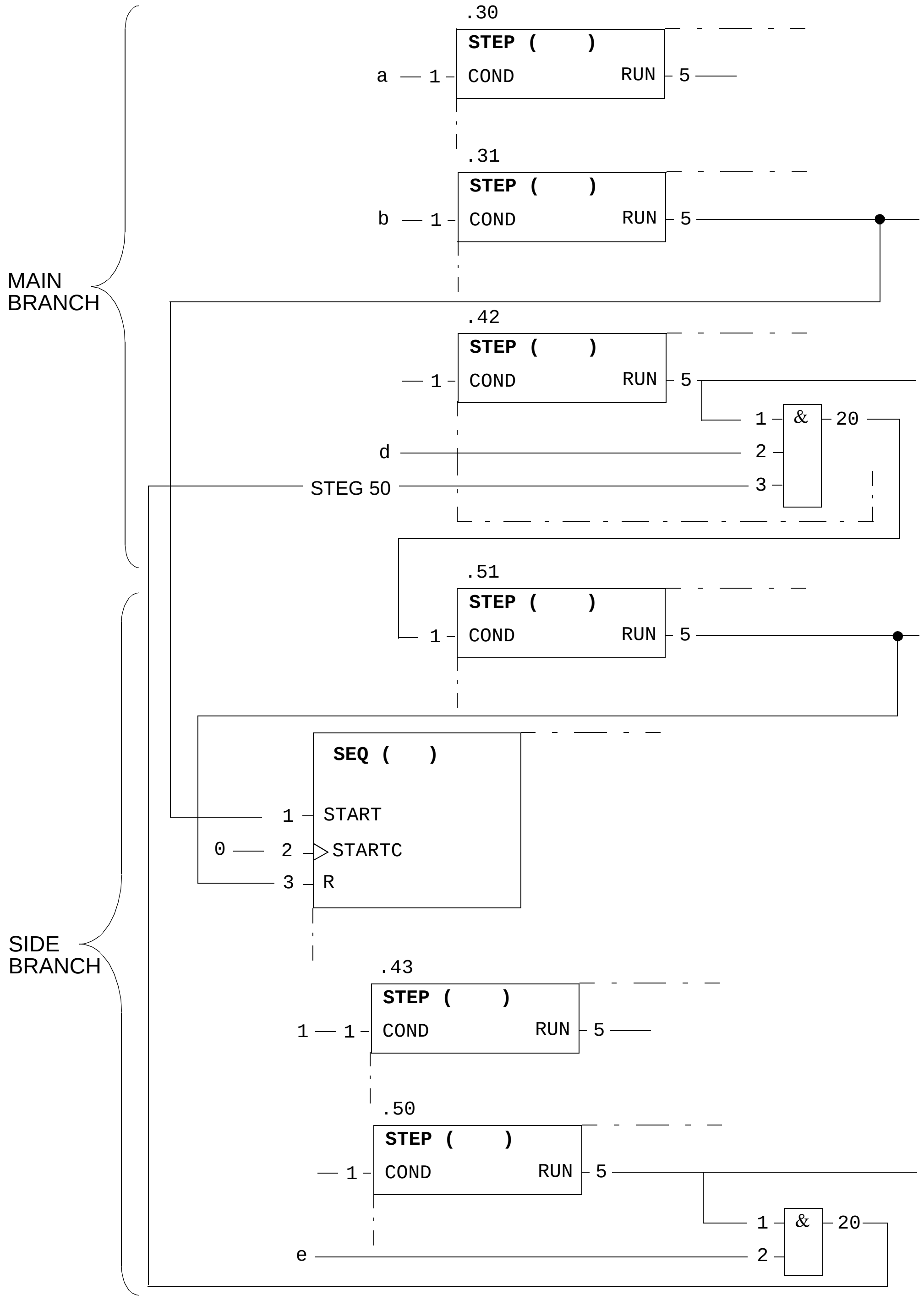


Figure 15-35. Implementation of Simultaneous Sequences according to Figure 15-34

15.9 Sequence with Loop

A sequence may contain a loop that is to be executed a number of times before the sequence continues. The loop in Figure 15-36 is to be executed n times. A test of the number of times the loop has been executed is performed in step 20. If all **n** executions are done the sequence proceeds to step 21. Otherwise the loop is executed again.

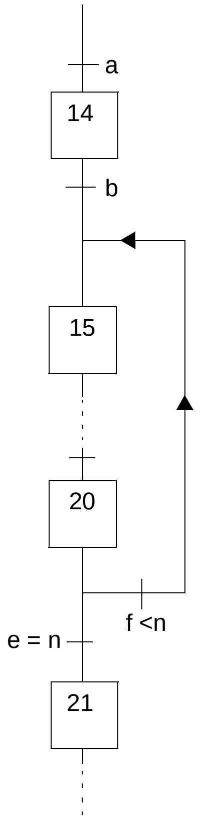


Figure 15-36. Sequence Loop

Figure 15-37 shows a solution. The counter in step 20 is preset to n and is decremented by one each time the step is executed.

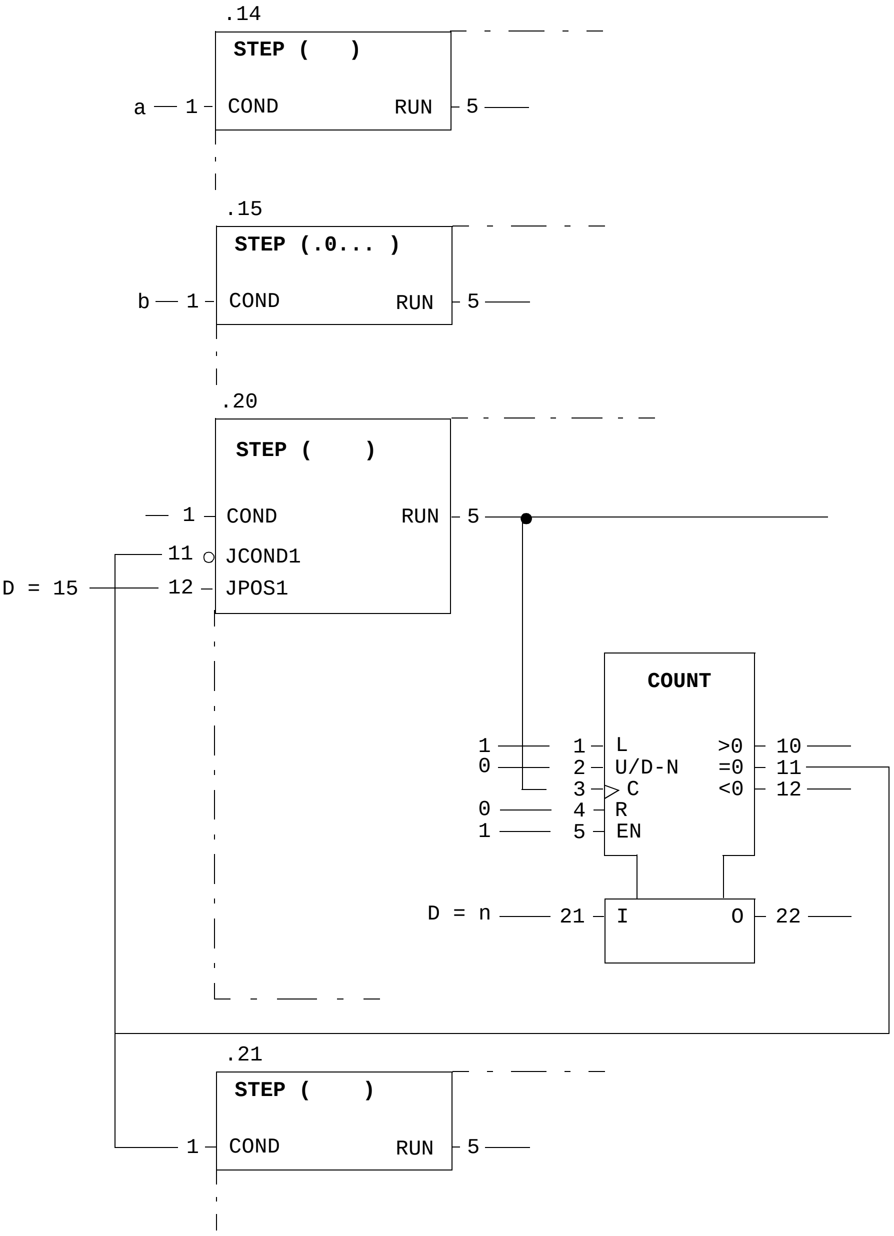


Figure 15-37. Implementation of Sequence Loop in Figure 15-36

15.10 Use of HOLD

The HOLD function described in Section 10.5.7 HOLD is used if one or more objects controlled by a sequence is temporarily to be controlled in another way.

Suppose that it is desired, in the example of the car sequence in Chapter 10, to be able to stop and reverse the car if an obstacle is encountered. Figure 15-38 shows a solution using the HOLD function. Control of the car is taken over by a module for manual control if the switch in the figure is placed in the HOLD position.

- It is normally not possible to manually control other activities than those of the active step without losing synchronization of the sequence since the execution of the step modules is stopped on HOLD.
- An object which may be manually controlled that is started during a sequence step must in a manual control module have a latch, for example in the way shown in Figure 15-38 (MOTOR M).

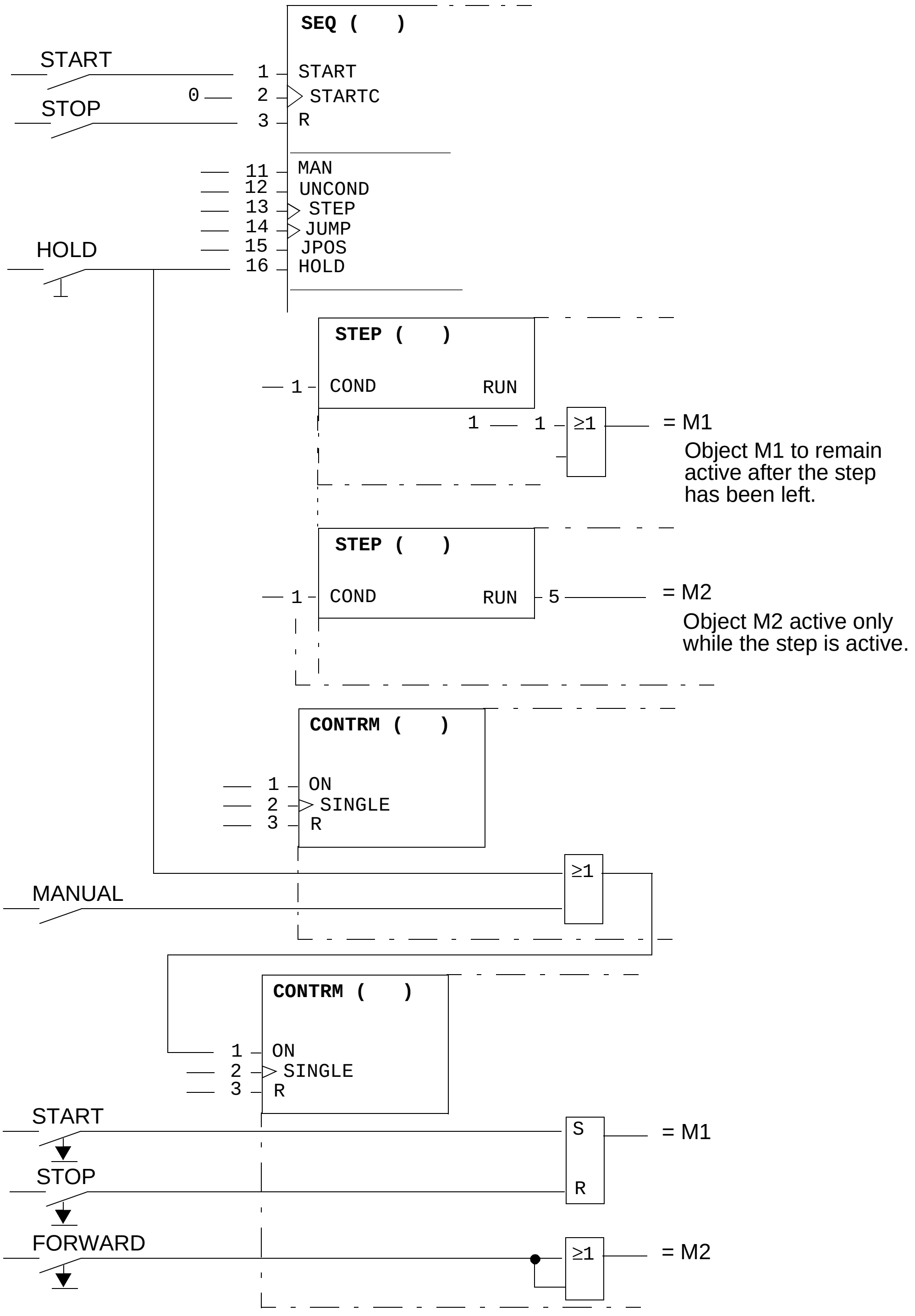


Figure 15-38. Example of Manual Control

Chapter 16 Application Example

As a conclusion we give an example of the procedure to prepare a PC program draft. Program entry, editing and documentation generation is not shown. Refer to the manual for the appropriate programming tools.

16.1 Control of Lift

The task is to control a passenger lift between two floors. The lift is powered by a motor with a forward/reverse switch as shown in Figure 16-1.

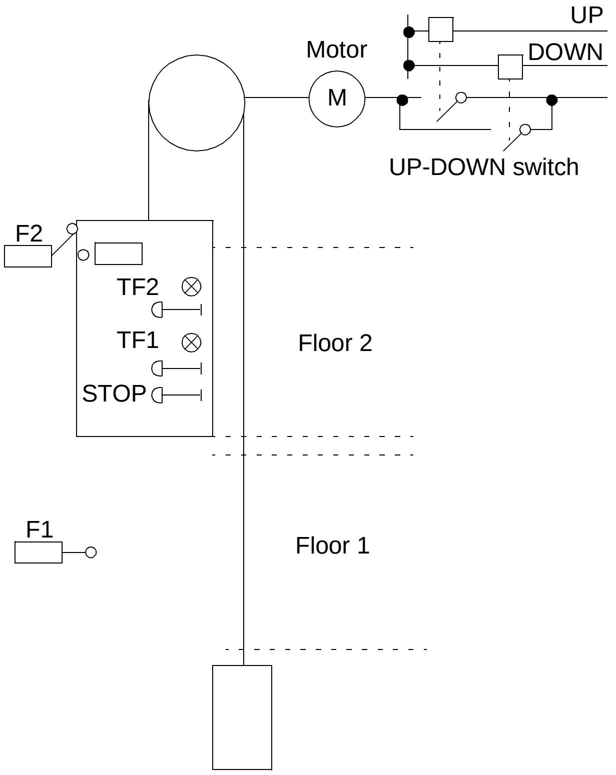


Figure 16-1. Example - Control of Lift

The lift is to be provided with supervision of over-load and running time as well as of the position of the door. The position of the lift carriage is detected with two position switches, one for each floor. To check that the door is closed another position switch is employed. To detect over-load the motor current is monitored.

The control equipment gives order to the forward/reverse switches for UP or DOWN.

One description of the process may be the block diagram in Figure 16-2, which shows the external signals that affect the lift control and the signals used to control the lift motor and to indicate operational status.

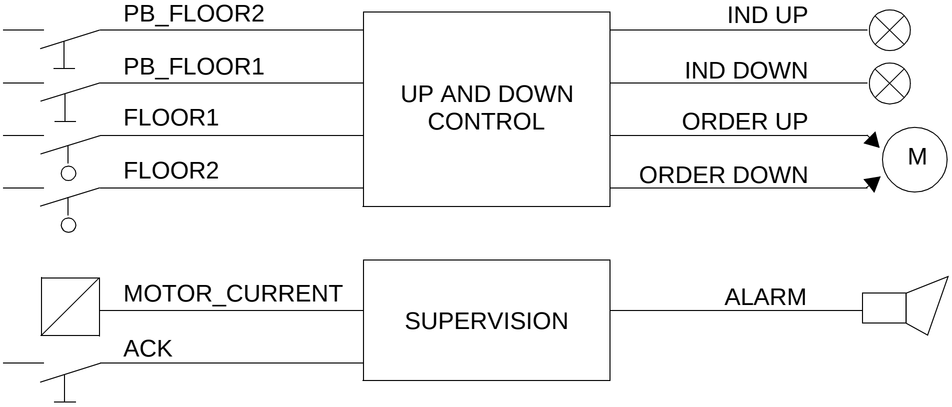


Figure 16-2. Block Diagram

The next operation is structuring the PC program, that is division of the program into PC modules and introducing module headers for execution control.

In this case a division into control modules for control and supervision is needed.

Having done this a logic diagram for the control equipment may be drawn. This may then appear as in Figure 16-3. Two timer circuits have been used for supervision. If the running time exceeds 4 minutes an alarm is triggered and the lift is stopped. Also, the lift is stopped and the alarm triggered if the motor current exceeds a certain value for more than 5 seconds.

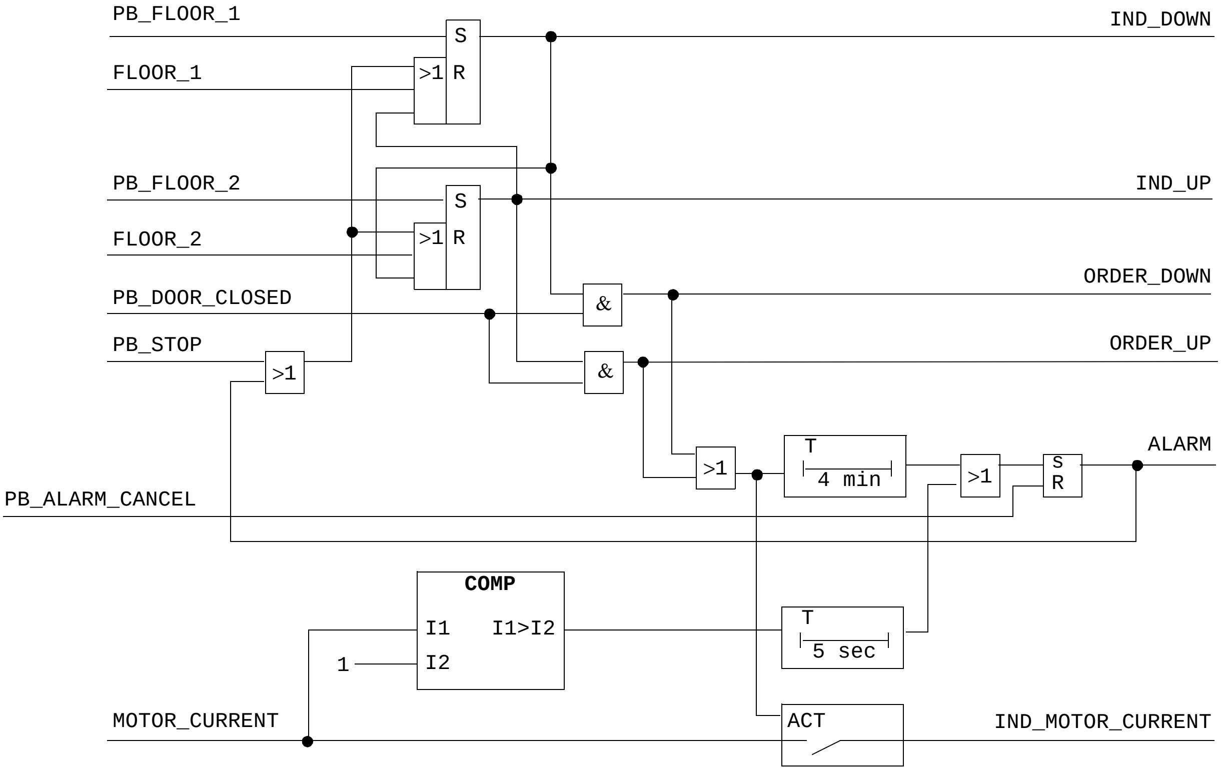


Figure 16-3. Logic Diagram for the Lift Control

This diagram specifies the desired function of the control equipment. It may now be converted into a draft PC diagram with PC elements chosen from the PC element library.

In this draft the variable names and operational parameters (time values) for the delay elements are entered.

If the circuit is analyzed from the execution point of view it can be seen that the control part must be executed with a short execution cycle for the stop signals to act sufficiently fast. The timing of the supervision part is not critical in the same way - for example the delay time for over-load alarm is set to 5 seconds. In Figure 16-4 the circuit has been divided into 2 PC modules, each with a control module header (CONTRM). The control part is to be given the cyclicity 20 ms and the supervision part 100 milliseconds.

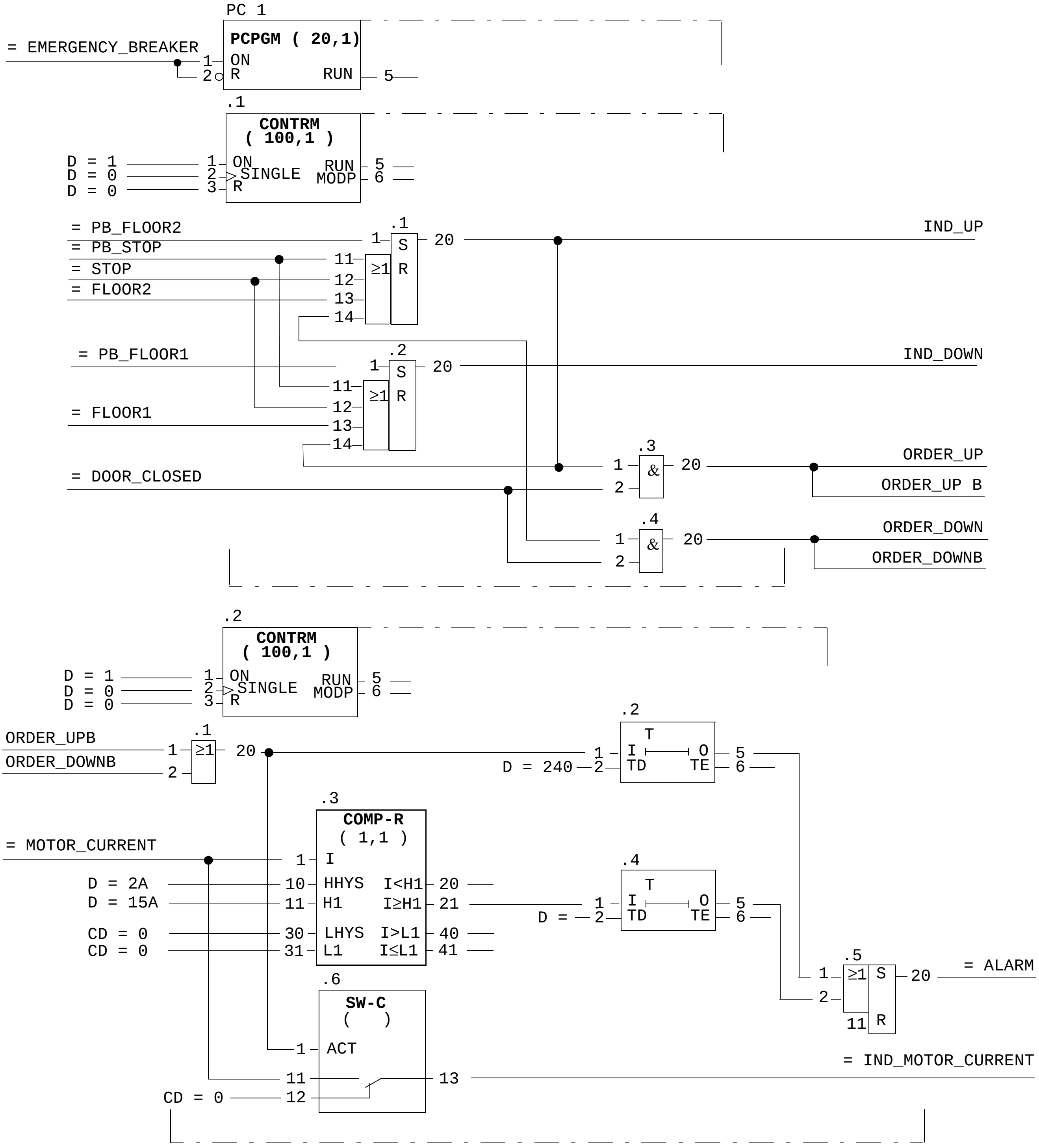


Figure 16-4. Draft PC Diagram with Two PC Modules

In the last step the PC modules are provided with a superior PC program according to Figure 16-5. The control header for the program (PCPGM) must always have a cyclicity belonging to interpreter A. In this case 20 milliseconds and place 1 in the table is chosen. The control part then gets place 2 in the same cycle time table. The supervision part gets place 1 in the table for 100 milliseconds. The superior PC program header will thus always be executed first.

Finally item designations for all PC modules and PC elements are entered and the program is ready for entry.

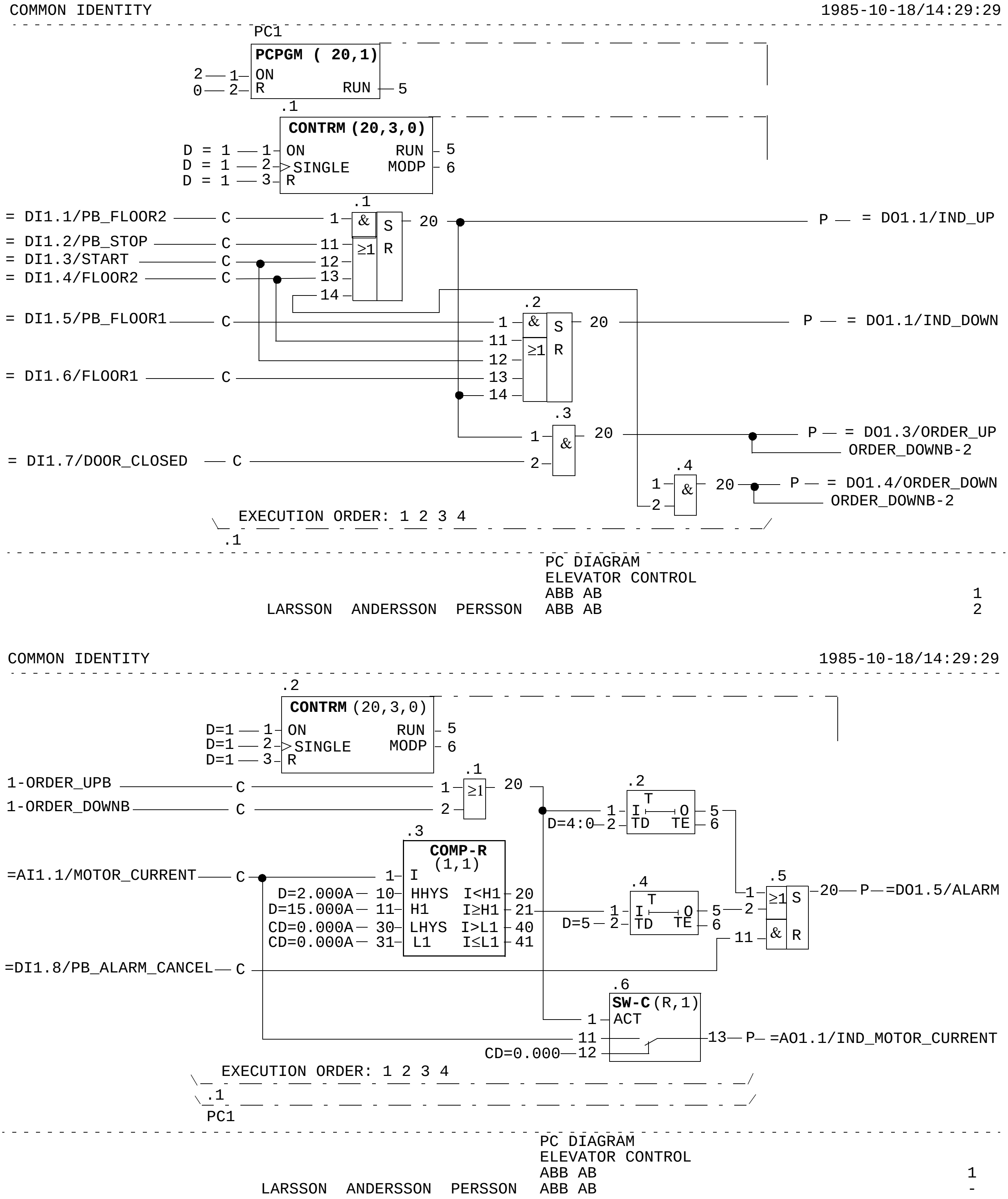


Figure 16-5. Completed PC Diagram

3BSE 003 841R0001

March, 1997